/thoughtworks

Strategy. Design. Engineering.

Volume 29 | Setembro 2023

Technology Radar

Um guia de opinião sobre o universo de tecnologia atual

Sobre o Radar

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

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

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

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

Radar em um relance

A ideia por trás do Radar é rastrear coisas interessantes, que chamamos de blips. Organizamos blips no Radar usando duas categorias: quadrantes e anéis. Os quadrantes representam as diferentes naturezas dos blips. Os anéis indicam nossa recomendação para utilizar a tecnologia.

Um blip é uma tecnologia ou técnica que desempenha um papel no desenvolvimento de software. Os blips estão "em movimento" - suas posições no radar estão constantemente mudando — geralmente indicando que nossa confiança em recomendá-los tem crescido à medida que se movimentam entre os anéis.

Adote: Acreditamos firmemente que a indústria deveria adotar esses itens. Nós os usamos quando são apropriados em nossos projetos.

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

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

Evite: Prossiga com cautela.

Novo **Mudança de anel** **Sem alterações**

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

Contribuições

O Conselho Consultivo de Tecnologia (em inglês, Technology Advisory Board - TAB) é um grupo formado por 22 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 nossa CTO Rachel Laycock, e a nossa CTO Emerita, Rebecca Parsons. O TAB atua examinando tópicos que afetam soluções de tecnologia e tecnologistas da Thoughtworks. Esta edição do Thoughtworks Technology Radar é baseada em uma reunião do TAB realizada remotamente em Agosto de 2023.

Rebecca Parsons (CTO Emerita)

Rachel Laycock (CTO)

Martin Fowler (Chief Scientist)

Bharani Subramaniam

Birgitta Böckeler

Brandon Byars

Camilla Falconi Crispim

Erik Dörnenburg

Fausto de la Torre

Hao Xu

Ian Cartwright

James Lewis

Marisa Hoenig

Maya Ormaza

Mike Mason

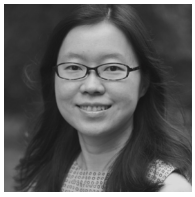
Neal Ford

Pawan Shah

Scott Shaw

Selvakumar Natesan

Shangqi Liu



Sofia Tania

Vanya Seth

Tradução: Akina Kurita, Augusto Barcellos Bernd, Bárbara Santos, Beatriz Oliveira, Camilla Crispim, Daiane Correa, Lais Gomes, Patrick Prado, Pietra Freitas, Renan Martins, Thiago Gregorio.

Temas

Desenvolvimento de Software assistido por IA

Não é surpresa para ninguém que os tópicos relacionados à IA dominaram nossa conversa para esta edição do Radar. Pela primeira vez, precisamos de um guia visual para desatar os nós das diferentes categorias e capacidades (algo que ainda não havíamos utilizado, mesmo no auge do caos no ecossistema JavaScript). Como uma consultoria de software com histórico de pioneirismo em práticas de engenharia inovadoras como CI e CD, uma das categorias de interesse para nós é o uso da IA para auxiliar o desenvolvimento de software. Como parte do Tech Radar, discutimos muitas ferramentas de assistência de programação como o GitHub Copilot, Tabnine e Codeium. Também estamos animadas com o potencial das LLMs de código aberto para agitar o cenário de ferramentas, e observamos uma grande promessa na explosão de ferramentas e recursos para assistência além da assistência na escrita de histórias das pessoas usuárias (user story, em inglês), pesquisa de pessoas usuárias, discurso de elevador e outras tarefas baseadas em linguagem. Ao mesmo tempo, esperamos que as pessoas desenvolvedoras usem todas essas ferramentas de forma responsável e permaneçam firmes no controle de elementos como as dependências alucinadas sendo este apenas um dos riscos de segurança e qualidade que você deve se atentar.

Quão produtivo é medir produtividade?

O desenvolvimento de software pode parecer magia para as não-tecnologistas, o que leva a gestão a se esforçar para medir o quão produtivas as pessoas desenvolvedoras são em suas tarefas misteriosas. Nosso cientista-chefe, Martin Fowler, escreveu sobre este tópico já em 2003, mas ele não desapareceu. Discutimos muitas ferramentas e técnicas modernas para este Radar que adotam abordagens mais sutis para medir o processo criativo de construção de software, mas ainda são insuficientes. Felizmente, a indústria se afastou do uso de linhas de código como indicador de produção. No entanto, maneiras alternativas de medir o A (“Atividade”) do framework SPACE, como o número de pull request ou problemas resolvidos, ainda são indicadores falhos de produtividade. Em vez disso, a indústria começou a se concentrar na engenharia efetiva: ao invés de medir a produtividade, devemos medir coisas que sabemos que contribuem ou prejudicam o fluxo. Ao contrário de nos concentrarmos nas atividades de um indivíduo, devemos nos concentrar nas fontes de desperdício do sistema e nas condições que podemos empiricamente mostrar que impactam na percepção de “produtividade” da pessoa desenvolvedora. Novas ferramentas como o DX DevEx 360 abordam isso focando na experiência da pessoa desenvolvedora, em vez de algum indicador incorreto de produção.

No entanto, muitos líderes continuam a se referir à “produtividade” das pessoas desenvolvedoras de forma vaga e qualitativa. Suspeitamos que pelo menos parte do ressurgimento deste interesse se justifique pela preocupação com o impacto do desenvolvimento de software assistido por IA, o que levanta a inevitável pergunta: é um impacto positivo? Embora as medições possam estar ganhando alguma nuance, as medições reais de produtividade ainda são ilusórias.

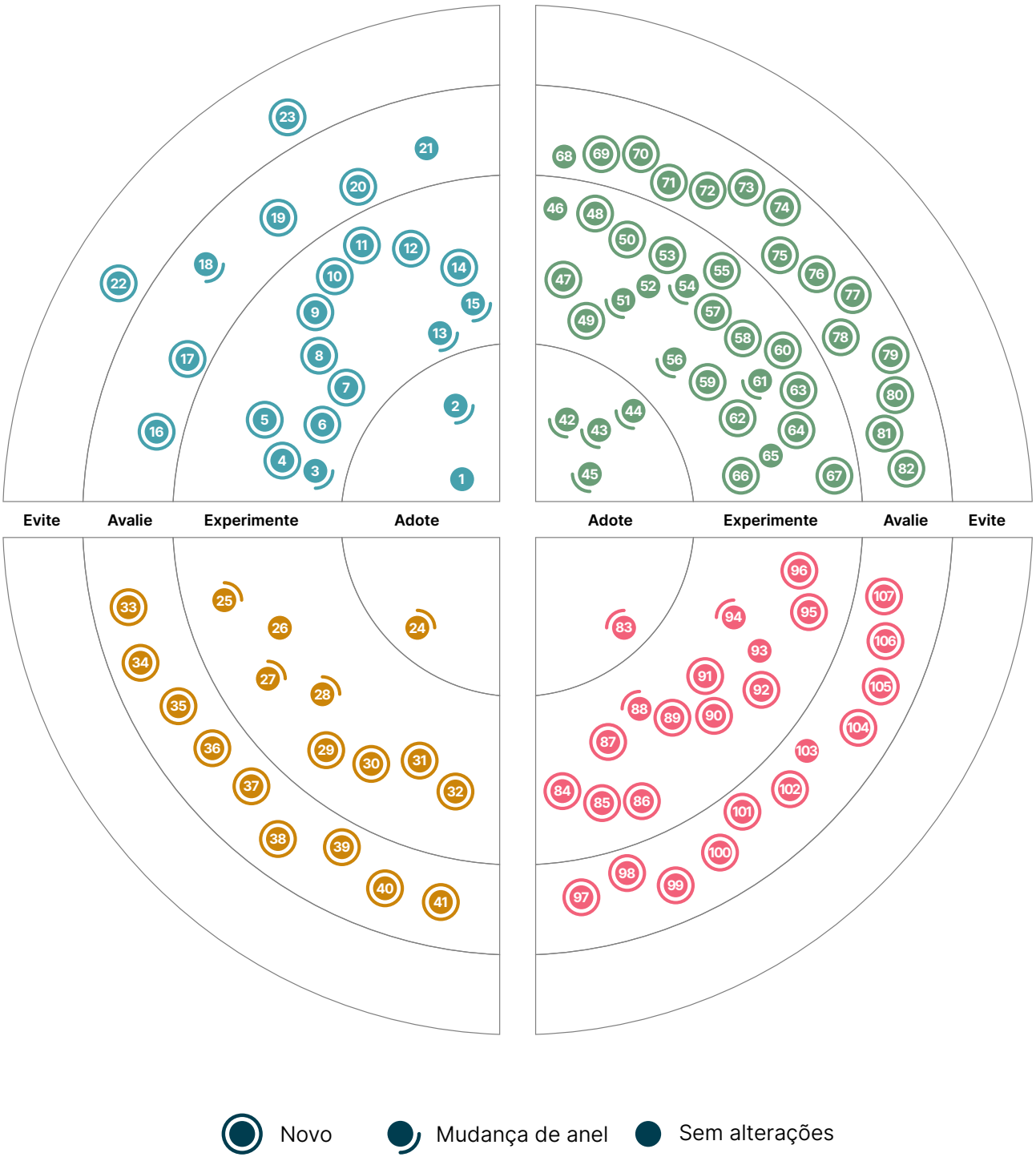
Uma grande quantidade de LLMs

Os modelos de linguagem de grande porte (LLMs) formam a base de muitos avanços modernos na IA. Muito da experimentação atual envolve acionar interfaces de usuárias semelhantes a bate-papo, como ChatGPT ou Bard. Fundamentalmente, os principais ecossistemas concorrentes (OpenAI's ChatGPT, Google's Bard, Meta's LLaMA, Amazon's Bedrock, entre outros) foram destaque em nossas discussões. De forma mais ampla, os LLMs são ferramentas que podem resolver uma variedade de problemas, desde a geração de conteúdo (texto, imagens e vídeos) à geração de código, resumo e tradução, para citar alguns. Com a linguagem natural servindo como uma poderosa camada de abstração, esses modelos apresentam um conjunto de ferramentas universalmente atraente e, portanto, estão sendo usados por muitas pessoas da área da informação. Nossa discussão abrange várias facetas dos LLMs, incluindo hospedagem própria, que permite personalização e maior controle do que os LLMs hospedados na nuvem. Com a crescente complexidade dos LLMs, deliberamos sobre a capacidade de quantificá-los e executá-los em pequenos formatos, especialmente em dispositivos de borda e ambientes com restrições. Mencionamos o ReAct prompting, que tem potencial para melhorar o desempenho, juntamente com agentes autônomos suportados por LLM que podem ser usados para construir aplicações dinâmicas que vão além das interações de pergunta e resposta. Também mencionamos vários bancos de dados vetoriais (incluindo Pinecone) que estão ressurgindo graças aos LLMs. As capacidades subjacentes dos LLMs, incluindo capacidades especializadas e hospedadas nativamente, continuam a crescer de forma acelerada.

O amadurecimento das soluções de entrega no trabalho remoto

Mesmo que equipes de desenvolvimento de software remotas tenham usado a tecnologia para superar restrições geográficas por anos, o impacto da pandemia impulsionou a inovação nessa área, solidificando o trabalho totalmente remoto ou híbrido como uma tendência duradoura. Para este Radar, discutimos como as práticas e ferramentas de desenvolvimento de software remoto amadureceram, e as equipes continuam a empurrar os limites com foco na colaboração eficaz em um ambiente mais distribuído e dinâmico do que nunca. Algumas equipes continuam a criar soluções inovadoras usando novas ferramentas colaborativas. Outras continuam a adaptar e melhorar as práticas presenciais existentes para atividades como a programação em par simultânea ou programação em grupo, workshops distribuídos (por exemplo, Event Storming remoto) e comunicação assíncrona e síncrona. Embora o trabalho remoto ofereça inúmeros benefícios (incluindo um pool de talentos mais diverso), o valor das interações presenciais é inquestionável. As equipes não devem deixar que ciclos críticos de feedbacks falhem e precisam estar cientes das concessões que assumem ao migrar para ambientes remotos.

O Radar



O Radar

Técnicas

Adote

1. Sistemas de design
2. Uma abordagem simples para RFCs

Experimente

3. Design de teste de componentes ciente da acessibilidade
4. Análise de caminho de ataque
5. Merge automático de PRs de atualização de dependência
6. Mentalidade de produto para dados FAIR
7. OIDC para GitHub Actions
8. Provisione monitores e alertas com Terraform
9. ReAct prompting
10. Retrieval-Augmented Generation (RAG)
11. Modelagem de falhas baseada em risco
12. Linguagem natural semiestruturada para LLMs
13. Rastreando saúde em vez de dívida técnica
14. Teste unitário para regras de alerta
15. Confiança zero para pipelines de CI/CD

Avalie

16. Verificações de saúde de dependência para combater alucinações de pacotes
17. Registros de decisão do sistema de design
18. GitOps
19. Agentes autônomos impulsionados por LLM
20. Sistemas de orquestração de plataforma
21. LLMs auto-hospedados

Evite

22. Ignorando as listas de top 10 da OWASP
23. Web Components para aplicações web renderizadas do lado do servidor (SSR)

Plataformas

Adote

24. Colima

Experimente

25. CloudEvents
26. DataOps.live
27. Google Cloud Vertex AI
28. Immuta
29. Lokalise
30. Orca
31. Trino
32. Wiz

Avalie

33. ActivityPub
34. Aplicativos de Contêiner do Azure
35. Serviço OpenAI do Azure
36. ChatGLM
37. Chroma
38. Kraftful
39. pgvector
40. Pinecone
41. wazero

Evite

—

Ferramentas

Adote

42. dbt
43. Mermaid
44. Ruff
45. Snyk

Experimente

46. AWS Control Tower
47. Bloc
48. cdk-nag
49. Checkov
50. Chromatic
51. Cilium
52. Cloud Carbon Footprint
53. Container Structure Tests
54. Devbox
55. DX DevEx 360
56. GitHub Copilot
57. Insomnia
58. Plugin HTTP Client do IntelliJ
59. KEDA
60. Kubeconform
61. mob
62. MobSF
63. Mocks Server
64. Prisma runtime defense
65. Terratest
66. Thanos
67. Yalc

Avalie

68. ChatGPT
69. Codeium
70. Fila de merges do GitHub
71. Google Bard
72. Google Cloud Workstations
73. Gradio
74. KWOK
75. Llama 2
76. Maestro
77. Modelos de linguagem de grande porte (LLMs) de código aberto para programação
78. OpenCost
79. OpenRewrite
80. OrbStack
81. Pixie
82. Tabnine

Evite

—

Linguagens e Frameworks

Adote

83. Playwright

Experimente

84. APIs Mínimas do .NET
85. Ajv
86. Armeria
87. AWS SAM
88. Dart
89. fast-check
90. Kotlin com Spring
91. Mockery
92. Netflix DGS
93. OpenTelemetry
94. Polars
95. Pushpin
96. Snowpark

Avalie

97. Perfis de Referência
98. GGML
99. GPTCache
100. API de Inflexão Gramatical
101. htmx
102. Kotlin Kover
103. LangChain
104. LlamaIndex
105. promptfoo
106. Semantic Kernel
107. Spring Modulith

Evite

—

Técnicas

Adote

1. Sistemas de design
2. Uma abordagem simples para RFCs

Experimente

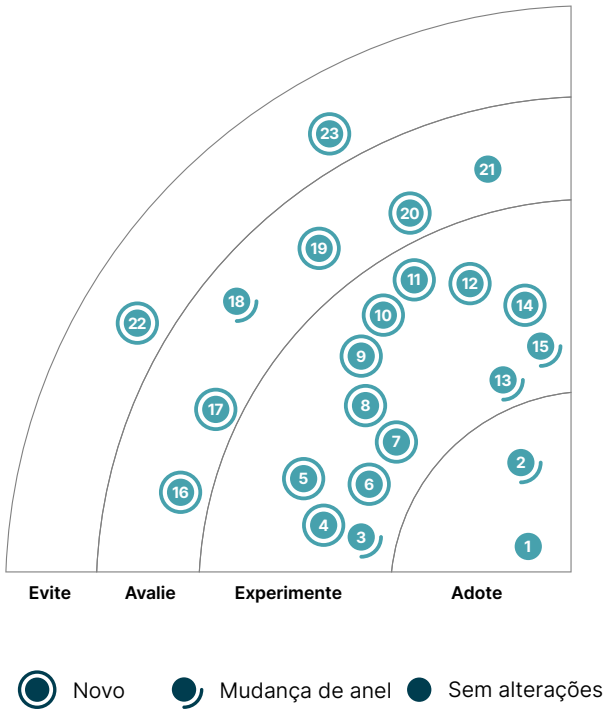
3. Design de teste de componentes ciente da acessibilidade
4. Análise de caminho de ataque
5. Merge automático de PRs de atualização de dependência
6. Mentalidade de produto para dados FAIR
7. OIDC para GitHub Actions
8. Provisione monitores e alertas com Terraform
9. ReAct prompting
10. Retrieval-Augmented Generation (RAG)
11. Modelagem de falhas baseada em risco
12. Linguagem natural semiestruturada para LLMs
13. Rastreando saúde em vez de dívida técnica
14. Teste unitário para regras de alerta
15. Confiança zero para pipelines de CI/CD

Avalie

16. Verificações de saúde de dependência para combater alucinações de pacotes
17. Registros de decisão do sistema de design
18. GitOps
19. Agentes autônomos impulsionados por LLM
20. Sistemas de orquestração de plataforma
21. LLMs auto-hospedados

Evite

22. Ignorando as listas de top 10 da OWASP
23. Web Components para aplicações web renderizadas do lado do servidor (SSR)



1. 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 é particularmente verdadeiro em organizações maiores com várias equipes trabalhando em produtos diferentes. **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. Uma evolução dos guias de estilo corporativos do passado, os sistemas de design oferecem bibliotecas e documentos compartilhados que são fáceis de encontrar e usar. Geralmente, as orientações são escritas em código e mantidas 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 ao trabalhar em equipes e disciplinas no desenvolvimento de produtos, pois permitem que as equipes se concentrem. Eles podem resolver desafios estratégicos em torno do produto em si sem reinventar a roda sempre que um novo componente visual for necessário.

Nossas experiências mostram que as equipes raramente aplicam uma mentalidade centrada no produto ao construir sistemas de design. Os principais consumidores das bibliotecas e documentos compartilhados são as equipes de desenvolvimento de produtos. Ao aplicar uma mentalidade de produto, as responsáveis pelo sistema devem estabelecer empatia com os grupos consumidores internos (as equipes de desenvolvimento) e colaborar com eles. Descobrimos que o motivo de muitas bibliotecas de componentes serem criticadas, é porque a equipe responsável não foi capaz de fornecer aos grupos consumidores o que eles precisavam de maneira rápida o suficiente e também não estavam configuradas para receber contribuições externas. Uma mentalidade centrada no produto também requer que as organizações pensem em como as contribuições devem ser feitas ao sistema de design e como essas contribuições devem ser governadas - neste tópico, recomendamos aplicar a técnica Registros de Decisões do Sistema de Design. Para nós, administrar um bom sistema de design ou biblioteca de componentes requer tanto trabalho social quanto técnico.

2. Uma abordagem simples para RFCs

Adote

O Request for Comments (RFC) é um documento formal que inclui ideias de design e arquitetura atreladas a um contexto para facilitar a colaboração e tomada de decisão das equipes. Quase todas as organizações que são nativas digitais e de alto crescimento usam RFCs para registrar as decisões sobre design, arquitetura, técnicas e as maneiras como suas equipes colaboram. Organizações maduras usam RFCs em equipes autônomas para promover uma melhor comunicação e colaboração, especialmente na tomada de decisão em conjunto. Eles são frequentemente usados como um processo para revisar e validar os registros de decisões de arquitetura. O resultado é um processo colaborativo e transparente que permite que as pessoas afetadas pela decisão tenham a chance de contribuir e compartilhar feedbacks antes que a decisão seja aprovada. Muitas vezes, em ambientes de rápida mudança, os motivos que levam à tomada de decisão em design se perdem ao longo do caminho, e as equipes responsáveis pela implementação dessa decisão ficam sem saber o que fazer. Um RFC fornece um registro de auditoria de decisão que beneficia os membros futuros da equipe e documenta a evolução técnica e comercial de uma organização. Um RFC pode ser uma ferramenta valiosa para facilitar a arquitetura evolutiva. Entretanto, para obter o melhor resultado, recomendamos **uma abordagem simples para RFCs**. Se não forem de escopo e objetivo limitados, esses documentos tendem a crescer em tamanho ao longo do tempo e começam a se assemelhar a documentos de arquitetura de solução tradicionais que são arquivados e esquecidos.

3. Design de teste de componentes ciente da acessibilidade

Experimente

Um dos muitos lugares no processo de entrega de software onde os requisitos de acessibilidade devem ser considerados é durante o teste de componentes web. Embora plugins de frameworks de teste como o chai-a11y-axe forneçam asserções em suas APIs para verificar o básico, **o design de teste de componentes ciente da acessibilidade** pode ajudar ainda mais a fornecer todos os elementos semânticos necessários para leitores de tela e outras tecnologias assistivas. Primeiro, ao invés de usar test ids ou classes para encontrar e selecionar os elementos que você deseja validar, utilize o princípio de identificar elementos por funções do ARIA ou outros atributos semânticos usados por tecnologias assistivas. Algumas bibliotecas de teste, como a Testing Library, até recomendam isso em sua documentação. Segundo, não realize testes apenas para interações por clique; considere também pessoas que não podem usar um mouse ou ver a tela e considere incluir testes adicionais para o teclado e outras interações. A técnica descrita está bem estabelecida em nossas equipes e deveríamos tê-la colocado no anel Experimente há algum tempo.

4. Análise de caminho de ataque

Experimente

A **análise de caminho de ataque** (ACA) é uma técnica de análise de segurança que identifica e avalia os potenciais caminhos que um invasor pode tomar para explorar vulnerabilidades nos sistemas e redes de uma organização. Anteriormente, a maioria das estratégias ou ferramentas de análise de segurança se concentrava em áreas de risco específicas, como configurações incorretas, contêineres vulneráveis ou alertas de CVE. Essa abordagem em silos significa que as equipes não podem ver como os riscos podem ser combinados com as fraquezas em outras camadas do stack de tecnologia para criar caminhos de ataque perigosos. Embora essa técnica não seja nova, os recentes avanços nas ferramentas de análise de segurança tornaram-na mais acessível às equipes de segurança. Orca e Wiz são duas dessas ferramentas. Sugerimos que as equipes que gerenciam infraestruturas complexas considerem essa técnica ao planejar uma estratégia de segurança ou selecionar as ferramentas de análise de segurança para sua organização.

5. Merge automático de PRs de atualização de dependência

Experimente

A complexidade da cadeia de suprimentos de software é um grande risco, e nós a abordamos extensivamente, por exemplo, em nossos artigos sobre SBOM e SLSA. O calcanhar de Aquiles para a maioria das equipes ainda é a presença de vulnerabilidades em dependências, muitas vezes dependências indiretas em vários níveis. Ferramentas como Dependabot ajudam criando pull requests (PRs) para atualizar as dependências. No entanto, é preciso disciplina de engenharia para cuidar dessas PRs de forma rápida, especialmente quando são para aplicativos ou serviços que não estão em desenvolvimento ativo. Nas circunstâncias certas, agora defendemos o uso do **merge automático de PRs de atualização de dependência.** Isso requer que o sistema tenha cobertura de teste extensiva — não apenas testes de unidade, mas também testes funcionais e de desempenho. A pipeline de compilação (build) deve executar todos esses testes e deve incluir a varredura de segurança. Em resumo, a equipe deve ter total confiança de que, quando a pipeline for executada com sucesso, o software estará pronto para ser colocado em produção. Nesses casos, as PRs de atualização de dependência, mesmo quando incluem atualizações de versão principal em dependências indiretas, devem ser mescladas (merged) automaticamente.

6. Mentalidade de produto para dados FAIR

Experimente

A mentalidade de produto para dados prioriza o tratamento das pessoas consumidoras de dados como clientes, garantindo que elas tenham uma experiência perfeita em toda a cadeia de valor dos dados. Isso abrange a facilidade de descoberta de dados, compreensão, confiança, acesso e consumo. A "mentalidade de produto" não é um conceito novo. No passado, nós o adotamos no mundo operacional ao construir produtos operacionais ou microsserviços. Ele também sugere uma nova maneira de construir equipes multifuncionais de longa duração para deter e compartilhar dados em toda a organização. Acreditamos que, ao trazer uma mentalidade de produto para os dados, as organizações podem operacionalizar os princípios FAIR (encontráveis, acessíveis, interoperáveis e reutilizáveis). Nossas equipes usam catálogos de dados como Collibra e DataHub para permitir a descoberta de produtos de dados. Para promover a confiança, publicamos métricas de qualidade de dados e SLI como recenticidade, completude e consistência para cada produto de dados, e ferramentas como Soda Core e Great Expectations automatizam as verificações de qualidade dos dados. A observabilidade de dados, por sua vez, pode ser obtida com a ajuda de plataformas como Monte Carlo. Temos visto os produtos de dados evoluírem como blocos de construção reutilizáveis para vários casos de uso ao longo do tempo. Isso é acompanhado por um tempo de lançamento mais rápido para casos de uso subsequentes à medida que avançamos na identificação e construção de produtos de dados orientados a valor. Portanto, nosso conselho é abraçar **a mentalidade de produto para dados FAIR.**

7. OIDC para GitHub Actions

Experimente

Uma das técnicas que recomendamos para implementar a confiança zero para pipelines de CI/CD é autenticar seus pipelines para acesso a serviços de nuvem por meio de mecanismos de identidade federada como OpenID Connect (OIDC). Como o GitHub Actions é amplamente utilizado — e essa técnica importante ainda é subutilizada — queremos destacar o **OIDC para GitHub Actions**. Dessa forma, você pode evitar o armazenamento de tokens de acesso de longa duração para seus recursos de nuvem e seus pipelines não terão acesso direto a segredos. No entanto, certifique-se de restringir o acesso cuidadosamente para que as ações realmente sejam executadas com privilégios mínimos.

8. Provisione monitores e alertas com Terraform

Experimente

Infraestrutura como código (IaC) é uma abordagem amplamente aceita para definir e provisionar ambientes de hospedagem. Mesmo com a contínua evolução de ferramentas e técnicas nessa área, o Terraform continua sendo a ferramenta dominante para fazer IaC em recursos nativos da nuvem. Entretanto, a maioria dos ambientes de hospedagem hoje são combinações complexas de serviços nativos de fornecedores de nuvem, serviços de terceiros e código personalizado. Nesses ambientes, percebemos que as pessoas engenheiras muitas vezes recorrem a uma mistura de Terraform para recursos de nuvem e scripts personalizados para o resto. Isso pode levar a uma falta de consistência e repetibilidade no processo de provisionamento. Na verdade, muitos dos serviços de terceiros que são comumente usados em ambientes de hospedagem — incluindo Splunk, Datadog, PagerDuty e New Relic — possuem provedores do Terraform que você pode usar para provisionar e configurar esses serviços. É por isso que recomendamos que, além de recursos de nuvem, as equipes também **provisionem monitores e alertas com Terraform.** Isso leva a uma IaC com melhor modularidade, que é mais fácil de entender e manter. Como em toda IaC, há um risco de introduzir inconsistências quando a configuração é alterada por meio de outras interfaces. Para garantir que o código do Terraform permaneça como fonte da verdade, recomendamos que você desabilite as alterações de configuração por meio de interfaces da pessoa usuária e APIs.

9. ReAct prompting

Experimente

O **ReAct prompting** é um método de prompt LLMs (modelos de linguagem de grande porte) que visa melhorar a precisão de suas respostas em relação a métodos concorrentes, como cadeia de pensamento (CoT). Introduzido em um artigo de 2022, ele funciona combinando raciocínio e ação (daí o nome ReAct). Tal abordagem ajuda a tornar as respostas dos LLMs mais explicáveis e reduz as alucinações em comparação com o CoT, dando as engenheiras de prompt uma chance melhor de obter o que desejam. LangChain foi originalmente desenvolvido para suportar esse estilo de prompt. Os agentes autônomos baseados no método de prompt ReAct provaram ser algumas das aplicações mais amplamente utilizadas de LLMs que nossas equipes têm construído. Recentemente, a OpenAI introduziu chamada de função em suas APIs para facilitar a implementação do ReAct e de estilos de prompt semelhantes sem recorrer a ferramentas externas como LangChain. Ainda estamos nos estágios iniciais da definição desta disciplina, mas até agora, ReAct e seus descendentes apontaram o caminho para algumas das aplicações mais fascinantes de LLMs.

10. Retrieval-Augmented Generation (RAG)

Experimente

Retrieval-Augmented Generation (RAG) é uma técnica para combinar memória paramétrica e não paramétrica pré-treinada para geração de linguagem. Ela permite que você aumente o conhecimento existente de LLMs pré-treinados com o conhecimento privado e contextual do seu domínio ou setor. Com RAG, você primeiro recupera um conjunto de documentos relevantes da memória não paramétrica (geralmente por meio de uma busca de similaridade a partir de um datastore vetorial) e, em seguida, usa a memória paramétrica dos LLMs para gerar uma saída que seja consistente com os documentos recuperados. Nós achamos que RAG é uma técnica eficaz para uma variedade de tarefas de processamento de linguagem natural (PLN) que requerem conhecimento profundo, incluindo respostas a perguntas, resumo e geração de histórias.

11. Modelagem de falhas baseada em risco

Experimente

Modelagem de falhas baseada em risco é um processo usado para entender o impacto, a probabilidade e a capacidade de detectar as várias maneiras que um sistema pode falhar. As equipes de entrega estão começando a usar essa metodologia para projetar e avaliar os controles necessários para prevenir essas falhas. A abordagem é baseada na prática da análise de modos e efeitos de falha (FMEA), uma técnica de avaliação de riscos que existe desde os anos 1940 e tem um histórico de sucesso em indústrias que constroem sistemas físicos complexos, como aeroespacial e automotivo. Assim como nessas indústrias, a falha de software também pode ter consequências graves — comprometendo, por exemplo, a saúde e a privacidade humana — razão pela qual estamos observando uma necessidade crescente dos sistemas serem submetidos a uma análise rigorosa. O processo começa a partir da identificação dos possíveis modos de falha. A equipe então realiza uma análise da causa raiz e atribui pontuações de acordo com a probabilidade de uma falha ocorrer, o tamanho de seu impacto e a probabilidade de detectar a causa raiz da falha. Descobrimos que isso é mais eficaz quando as equipes multifuncionais iteram por esse processo à medida que o sistema evolui. No que diz respeito à segurança, modelagem de falhas baseada em risco pode ser um complemento útil à modelagem de ameaças e à análise de caminho de ataque.

12. Linguagem natural semiestruturada para LLMs

Experimente

Temos tido sucesso em várias aplicações usando **linguagem natural semiestruturada para LLMs.** As entradas estruturadas, como um documento JSON, são nítidas, precisas e dão ao modelo uma indicação do tipo de resposta que está sendo buscada. Restringir a resposta dessa maneira ajuda a estreitar o espaço do problema e completar de forma mais precisa, particularmente quando a estrutura está em conformidade com uma linguagem de domínio específico (DSL) cuja sintaxe ou esquema é fornecido ao modelo. Também descobrimos que o enriquecimento da entrada estruturada com comentários ou notações de linguagem natural produz uma resposta melhor do que a linguagem natural ou a entrada estruturada sozinha. Tipicamente, a linguagem natural é simplesmente intercalada com conteúdo estruturado ao construir o prompt. Como em muitos comportamentos de LLM, não sabemos exatamente por que isso funciona, mas nossa experiência mostra que colocar comentários de linguagem natural no código escrito por humanos também melhora a qualidade da saída para assistentes de programação baseados em LLM.

13. Rastreando saúde em vez de dívida técnica

Experimente

Estamos observando que as equipes estão melhorando seu ecossistema ao tratar a classificação de saúde da mesma forma que outros objetivos de nível de serviço (SLO) e priorizando as melhorias de acordo, em vez de se concentrarem apenas no rastreamento da dívida técnica. Ao alocar recursos de forma eficiente para resolver os problemas de maior impacto relacionados à saúde, as equipes e organizações podem reduzir os custos de manutenção de longo prazo e evoluir os produtos de forma mais eficiente. Essa abordagem também aprimora a comunicação entre stakeholders técnicos e não técnicos, promovendo uma compreensão comum do estado do sistema. Embora as métricas possam variar entre as organizações (veja este post do blog para exemplos), elas contribuem para a sustentabilidade de longo prazo e garantem que o software permaneça adaptável e competitivo. Em um cenário digital em constante mudança, **rastrear a saúde em vez da dívida técnica** dos sistemas fornece uma estratégia estruturada e baseada em evidências para mantê-los e melhorá-los.

14. Teste unitário para regras de alerta

Experimente

Observabilidade e monitoramento são essenciais para equipes de software. Dada a natureza imprevisível de certos eventos, é crucial criar mecanismos de alerta precisos com regras complexas. No entanto, a verdadeira validação dessas regras só ocorre quando os cenários surgem em produção. A técnica de **teste unitário para regras de alerta** permite que as equipes definam melhor as regras testando-as e refinando-as proativamente antes, aumentando a confiança na forma como a regra está configurada. Isso ajuda a reduzir os alarmes falsos e garantir que os problemas reais sejam destacados. Ferramentas como Prometheus suportam teste unitário para regras; nossas equipes já estão relatando seus benefícios em cenários do mundo real.

15. Confiança zero para pipelines de CI/CD

Experimente

Se não forem devidamente protegidas, a infraestrutura e as ferramentas que executam nossas pipelines de compilação e de entrega podem se tornar uma grande vulnerabilidade. Os pipelines precisam acessar dados e sistemas críticos, como código, credenciais e segredos, para compilar e implantar software. Isso torna esses sistemas muito atrativos para agentes maliciosos. Portanto, recomendamos fortemente a aplicação da arquitetura de **confiança zero para pipelines de CI/CD** e infraestrutura

— confiando neles o mínimo necessário. Isso engloba uma série de técnicas: se disponível, faça a autenticação dos seus pipelines com seu provedor de nuvem por meio de mecanismos de identidade federada como OIDC; em vez de dar a eles acesso direto a segredos, implemente o princípio do menor privilégio, minimizando o acesso de contas de usuárias individuais ou de execução, em vez de empregar contas de acesso ilimitado; use seus executores de forma efêmera em vez de reutilizá-los, para reduzir o risco de expor segredos de atividades anteriores ou executar atividades em executores comprometidos; mantenha o software em seus agentes e executores atualizado, monitore a integridade, confidencialidade e disponibilidade de seus sistemas CI/CD da mesma forma que você monitora seu software em produção.

Estamos observando equipes se esquecendo desse tipo de prática, principalmente quando estão acostumadas a trabalhar com uma infraestrutura autogerenciada de CI/CD em zonas de rede internas. Embora todas essas práticas sejam importantes em suas redes internas, elas se tornam ainda mais cruciais ao usar um serviço gerenciado, pois isso amplia ainda mais a superfície de ataque e o raio de impacto.

16. Verificações de saúde de dependência para combater alucinações de pacotes

Avalie

A segurança da cadeia de suprimentos de software tornou-se uma preocupação comum entre as equipes de entrega, refletida no crescente número de ferramentas e técnicas no espaço, várias das quais já abordamos anteriormente no Radar. A crescente popularidade de ferramentas baseadas em GenAI como auxiliares no processo de desenvolvimento de software introduziu um novo vetor de ataque à cadeia de suprimentos de software: alucinações de pacotes. Acreditamos que é importante que as equipes que usam essas ferramentas GenAI em seu processo de desenvolvimento fiquem vigilantes contra esse risco. Para garantir isso, as equipes podem realizar **verificações de saúde de dependências para combater alucinações de pacotes:** olhar a data de criação, o número de downloads, os comentários e as notas atribuídas, número de colaboradores, histórico de atividade e assim por diante antes da decisão final de adoção da ferramenta. Algumas dessas verificações podem ser realizadas em repositórios de pacotes e no GitHub, e ferramentas como deps.dev e Snyk advisor que também podem fornecer informações adicionais. Embora não seja uma técnica nova, ela está ganhando de volta a sua relevância à medida que as equipes experimentam cada vez mais ferramentas GenAI em seu processo de desenvolvimento de software.

17. Registros de decisão do sistema de design

Avalie

Em um ambiente de desenvolvimento de produto em ritmo acelerado, onde as necessidades das pessoas usuárias estão em constante evolução, o design é uma área que está sempre em desenvolvimento. Isso significa que os inputs sobre as decisões de design continuarão a ser necessários. Adotando a ideia de documentar decisões de arquitetura por meio de ADRs (Architecture Decision Records), começamos a adotar um formato semelhante — **registros de decisão do sistema de design** — para documentar decisões do sistema de design com a respectiva justificativa, percepções de pesquisa e resultados de experimentos. Comunicar decisões do sistema de design de forma eficaz parece ser uma necessidade emergente das equipes de desenvolvimento de produtos; fazer isso de maneira leve também é recomendado pelo zeroheight. Essa técnica nos ajudou a reduzir o tempo de onboarding, a avançar as conversas e a alinhar os fluxos de trabalho que compartilham o mesmo sistema de design.

18. GitOps

Avalie

GitOps é uma técnica de implantação de aplicações usando o padrão control loop. Um operador mantém a aplicação implantada sincronizada com a configuração, geralmente um repositório Git. Quando escrevemos pela última vez sobre GitOps, a comunidade ainda não havia chegado a um acordo sobre uma definição do termo. Na época, estávamos preocupadas com interpretações comuns da técnica que incluíam abordagens como branch por ambiente para configuração, o que pode levar a snowflakes como código. Além disso, a mensagem sobre GitOps como uma alternativa à entrega contínua era confusa. Desde então, os quatro princípios do GitOps especificam o escopo e a natureza da técnica. Quando você retira a expectativa exagerada e a confusão, GitOps é uma técnica útil que aproveita a funcionalidade de um cluster Kubernetes e cria oportunidades para separação de conceitos entre a configuração de uma aplicação e a implementação do processo de implantação. Algumas de nossas equipes implementaram o GitOps como parte da estruturação do seu processo de entrega contínua e tiveram experiências positivas, razão pela qual recomendamos avaliá-lo.

19. Agentes autônomos impulsionados por LLM

Avalie

À medida que o desenvolvimento de modelos de linguagem de grande porte (LLMs) continua, o interesse em construir agentes de IA autônomos é forte. AutoGPT, GPT-Engineer e BabyAGI são todos exemplos de **agentes autônomos impulsionados por LLM** que fomentam um LLM subjacente para entender o objetivo que lhes foi dado e trabalhar para ele. O agente lembra de quanto progresso fez, usa o LLM para raciocinar sobre o que fazer a seguir, toma ações e entende quando o objetivo foi alcançado. Isso é frequentemente conhecido como raciocínio de cadeia de pensamento — e pode realmente funcionar. Uma de nossas equipes implementou um chatbot de atendimento ao cliente como um agente autônomo. Se o bot não conseguir atingir o objetivo da cliente, ele reconhece sua própria limitação e redireciona a cliente para um humano. Essa abordagem está definitivamente no início de seu ciclo de desenvolvimento: agentes autônomos frequentemente sofrem de uma alta taxa de falha e incorrem em taxas caras de serviços de IA, e pelo menos uma startup de IA se afastou de uma abordagem baseada em agente.

20. Sistemas de orquestração de plataforma

Avalie

Com a adoção generalizada da engenharia de plataforma, estamos vendo uma nova geração de ferramentas que vão além do modelo tradicional de plataforma como serviço (PaaS) e oferecem contratos publicados entre equipes de plataforma e pessoas desenvolvedoras. O contrato pode envolver o provisionamento de ambientes de nuvem, bancos de dados, monitoramento, autenticação e muito mais em um ambiente diferente. Essas ferramentas garantem os padrões organizacionais enquanto concedem às pessoas desenvolvedoras acesso self-service a variações por meio de configuração. Exemplos desses **sistemas de orquestração de plataforma** incluem Kratix e Humanitec Platform Orchestrator. Recomendamos que as equipes de plataforma avaliem essas ferramentas como uma alternativa a reunir sua própria coleção única de scripts, ferramentas nativas e infraestrutura como código. Também notamos uma semelhança com os conceitos do Open Application Model (OAM) e seu orquestrador de referência KubeVela, embora o OAM afirme ser mais focado na aplicação do que no workload.

21. LLMs auto-hospedados

Avalie

Modelos de linguagem de grande porte (LLMs) geralmente requerem infraestrutura significativa de GPU para operar, porém há uma forte imposição para fazê-los funcionar em um hardware mais modesto. A quantização de um modelo de grande porte pode reduzir os requisitos de memória, permitindo que um modelo de alta fidelidade seja executado em hardware de custo menor ou até mesmo em uma CPU. Esforços como o llama.cpp tornam possível executar LLMs em hardware, incluindo Raspberry Pis, laptops e servidores de commodities. Muitas organizações estão implantando **LLMs auto-hospedados.** Isso geralmente ocorre devido a preocupações de segurança ou privacidade, ou, às vezes, à necessidade de executar modelos em dispositivos de borda. Exemplos de código aberto incluem GPT-J, GPT-JT, e Llama. Essa abordagem oferece melhor controle do modelo durante o ajuste fino para um caso de uso específico, segurança e privacidade aprimoradas, bem como acesso offline. Embora tenhamos ajudado alguns de nossos clientes a hospedar LLMs de código aberto para completar código, recomendamos que você avalie cuidadosamente as capacidades organizacionais e o custo de executar esses LLMs, antes de tomar a decisão de hospedá-los.

22. Ignorando as listas de top 10 da OWASP

Evite

O top 10 da OWASP é há muito tempo uma referência de consulta para os riscos de segurança mais críticos para aplicações web. Apesar de ser bem conhecida, já escrevemos sobre ela ser subutilizada no processo de desenvolvimento de software e alertamos contra ignorar o top 10 da OWASP.

O que não é muito conhecido é que a OWASP também publica listas semelhantes de top 10 para outras categorias. A lista de top 10 da OWASP para LLMs, cuja primeira versão foi lançada no início de agosto, destaca riscos como injeção de prompt, manipulação insegura de saída, envenenamento de dados de treinamento e outras que as pessoas desenvolvedoras e as equipes que criam aplicações LLM devem estar cientes. A OWASP também lançou recentemente a segunda versão da sua lista de top 10 da OWASP para API. Dada a amplitude de cobertura, qualidade e relevância das listas de top 10 da OWASP (aplicações web, APIs, LLMs e mais) para o panorama de segurança em constante mudança, estendemos nossa recomendação anterior e alertamos as equipes contra **ignorar as listas de top 10 da OWASP.**

23. Web Components para aplicações web renderizadas do lado do servidor (SSR)

Evite

Desde que os mencionamos pela primeira vez em 2014, os Web Components se tornaram populares e, em geral, nossa experiência tem sido positiva. Da mesma forma, defendemos a renderização de HTML no servidor, alertando contra SPA por padrão e incluindo frameworks como Next.js e htmx além de frameworks tradicionais de back-end. No entanto, embora seja possível combinar ambos, isso também pode se mostrar profundamente problemático. É por isso que sugerimos evitar **Web Components para aplicações web renderizadas do lado do servidor (SSR)**. Por ser uma tecnologia de navegador, não é trivial usar Web Components no servidor. Frameworks surgiram para tornar isso mais fácil, às vezes até usando um motor de navegação (browser engine), mas a complexidade ainda existe. Pior do que os problemas para a experiência da pessoa desenvolvedora é a experiência da pessoa usuária: o tempo de carregamento da página é impactado quando os Web Components personalizados precisam ser carregados e renderizados no navegador, e mesmo com pré-renderização e ajustes cuidadosos do componente, um flash de conteúdo sem estilo ou algum deslocamento de layout é quase inevitável. A decisão de abster-se de Web Components pode ter consequências abrangentes, como uma de nossas equipes comprovou quando precisou mover seu sistema de design para longe do Stencil, que é baseado em Web Components.

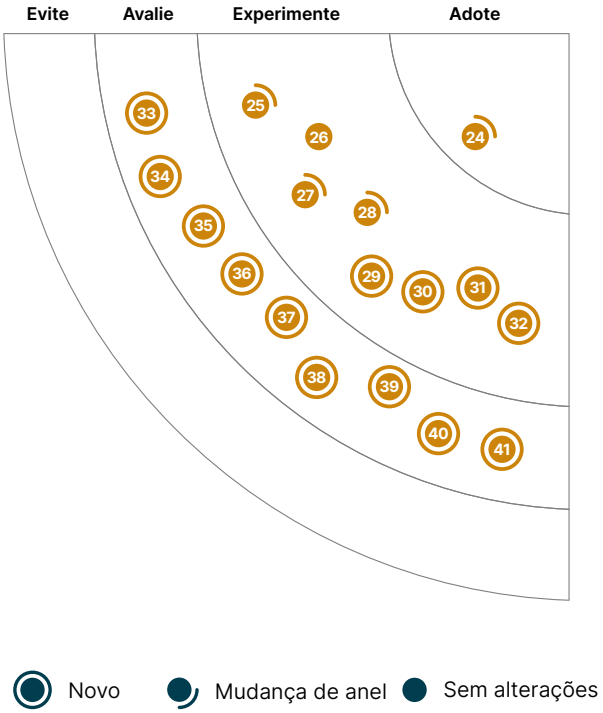
Plataformas

Adote
24. Colima

Experimente
25. CloudEvents
26. DataOps.live
27. Google Cloud Vertex AI
28. Immuta
29. Lokalise
30. Orca
31. Trino
32. Wiz

Avalie
33. ActivityPub
34. Aplicativos de Contêiner do Azure
35. Serviço OpenAI do Azure
36. ChatGLM
37. Chroma
38. Kraftful
39. pgvector
40. Pinecone
41. wazero

Evite
—



24. Colima

Adote

Colima é agora a nossa alternativa preferida ao Docker Desktop no macOS. Continuamos a usá-lo em vários projetos para provisionar o runtime do contêiner Docker em uma VM Lima, para configurar CLI do Docker no macOS e para lidar com o encaminhamento de porta e montagens de volume. O Colima pode ser configurado para executar o containerd como seu runtime, que também é o runtime da maioria dos serviços Kubernetes gerenciados, melhorando a importante equivalência entre os ambientes de desenvolvimento e produção.

25. CloudEvents

Experimente

Eventos são mecanismos comuns em arquiteturas orientadas a eventos ou aplicações sem servidor. No entanto, produtores ou provedores de nuvem tendem a apoiá-los em diferentes formatos, o que impede a interoperabilidade entre plataformas e infraestruturas. **CloudEvents** é uma especificação para descrever dados de eventos em formatos comuns para fornecer interoperabilidade entre serviços, plataformas e sistemas. Ele fornece SDKs em várias linguagens para que você possa incorporar a especificação em seu aplicativo ou conjunto de ferramentas. Nossas equipes o usam para fins de plataforma de multi-nuvem e também para especificação de eventos de domínio, entre outros cenários. CloudEvents é hospedado pela Cloud Native Computing Foundation (CNCF) e agora é executado como um projeto de incubação que vem ganhando cada vez mais atenção da indústria.

26. DataOps.live

Experimente

DataOps.live é uma plataforma de dados que automatiza ambientes no Snowflake. Inspirada nas práticas de DevOps, DataOps.live permite tratar a plataforma de dados como qualquer outra plataforma web, adotando a integração contínua e a entrega contínua (CI/CD), testes automatizados, observabilidade e gerenciamento de código. Nossas equipes estão usando-a para gerenciar o ciclo de vida de produtos de dados, incluindo desenvolvimento, branching e implantação de código e dados. Com seu gerenciamento de ambiente automatizado, é muito fácil criar ambientes com base em feature branches, modificá-los e destruí-los automaticamente. Também vale a pena notar sua capacidade de especificação declarativa (SOLE), que permite uma experiência simplificada para pessoas desenvolvedoras. Isso permite que as equipes reduzam o tempo necessário para construir produtos de dados de meses para dias. Nossas equipes têm usado DataOps.live com sucesso em produção, e é por isso que recomendamos esta plataforma ao trabalhar com Snowflake.

27. Google Cloud Vertex AI

Experimente

Ocorreram desenvolvimentos significativos no cenário de IA desde que vimos pela primeira vez o **Google Cloud Vertex AI**. De maio de 2023 até o momento, o Google lançou vários serviços e recursos para enriquecer esse campo. Essas adições incluem Model Garden: um repositório de mais de 100 modelos pré-treinados; Generative AI Studio: um console para explorar e prototipar rapidamente modelos de IA generativa; Vertex AI Extensions, que fornece ferramentas de desenvolvimento totalmente gerenciadas para conectar modelos de IA e dados em tempo real ou ações por meio de APIs. A plataforma evoluiu para oferecer modelos GenAI e suporte à integração, e estamos animadas para usá-la de forma mais extensiva.

28. Immuta

Experimente

Desde a última vez que escrevemos sobre o **Immuta**, nossas equipes ganharam experiência significativa com essa plataforma de segurança de dados. Seus destaques incluem a capacidade de definir políticas de assinatura e dados como código, controle de versão e a capacidade de implantar essas políticas automaticamente em ambientes superiores. Seu suporte ABAC nos permite associar tags a fontes de dados; se a mesma tag estiver associada à pessoa usuária, o acesso é concedido. Ao aproveitar a integração do Immuta e do Snowflake, foi possível automatizar a concessão de acesso a produtos de dados ou conjuntos de dados de forma autosuficiente. Quando a "pessoa usuária" solicita acesso a um produto de dados ou a um conjunto de dados, a tag do produto de dados é então associada à "pessoa usuária" como um atributo após a aprovação. Como o atributo da "pessoa usuária" corresponde à tag na fonte de dados, o acesso é concedido automaticamente por cortesia da política de Assinatura Global do Immuta. Também vale a pena notar as políticas de mascaramento de dados do Immuta, que preservam a privacidade dos dados mascarando e restringindo informações de PII a uma pessoa usuária específica. O acesso adicional a informações sensíveis, em um nível muito mais granular, pode ser definido usando políticas de segurança de nível de linha, que garantem que as pessoas usuárias só tenham acesso aos dados específicos aos quais estão autorizados a visualizar. Estamos satisfeitas com o Immuta, razão pela qual a estamos movendo para o Experimente - ela oferece uma boa experiência para as pessoas desenvolvedoras e torna mais fácil o gerenciamento das políticas de dados em grandes organizações.

29. Lokalise

Experimente

Lokalise é uma plataforma de localização totalmente automatizada que permite traduções contextuais. Nossos times usam a API do Lokalise em seus pipelines ETL ou fluxos de trabalho de desenvolvimento para traduzir informações localizáveis. O Lokalise suporta vários formatos de arquivo para strings localizáveis. Um aspecto a destacar é a capacidade de fazer o upload de um arquivo inteiro, onde cada par chave-valor é tratado como um registro separado e traduzido. Por baixo dos panos, aproveitamos a integração do Lokalise com o Google MT para cuidar das traduções. A interface web do Lokalise oferece facilidade de acesso a revisores humanos para verificar as traduções, encurtá-las e reescrevê-las conforme for adequado. No passado, destacamos ferramentas semelhantes, como Phrase. Nossos times tiveram uma boa experiência com Lokalise e recomendamos que você avalie a plataforma para fluxos de trabalho de tradução colaborativa.

30. Orca

Experimente

Orca é uma plataforma de segurança de nuvem proprietária que identifica, prioriza e corrige problemas de segurança e conformidade. Ela suporta os principais provedores de nuvem e configurações híbridas. Orca possui consultas/regras de segurança extensas para monitorar continuamente as cargas de trabalho implantadas quanto a configuração incorretas, vulnerabilidades e problemas de conformidade. Ela suporta VMs na nuvem, funções sem servidor, contêineres e aplicações Kubernetes para as cargas de trabalho implantadas. Essas regras de segurança embutidas são atualizadas constantemente para acompanhar as normas de conformidade e vetores de ameaças em evolução. Como Orca é sem agente, ela oferece uma boa experiência para as pessoas desenvolvedoras e é fácil de configurar. Outro recurso notável é que ela facilita a segurança desde o início. Nossas equipes usam Orca CLI para escanear imagens de contêiner e modelos de IaC quanto a vulnerabilidades e configuração incorretas como um hook de pré-commit ou como parte de fluxos de trabalho CI/CD. Ela também monitora e escanea continuamente repositórios de contêineres (por exemplo, AWS ECR) em busca de imagens base vulneráveis ou dependências do SO fracas para imagens já publicadas. Com base nas experiências das nossas equipes, Orca fornece uma visão unificada do estado de segurança ao longo do caminho para a produção, e por esse motivo a colocamos em Experimente.

31. Trino

Experimente

Trino, anteriormente conhecido como PrestoSQL, é um motor de consulta SQL distribuído e de código aberto, projetado para consultas analíticas interativas em big data. É otimizado para ser executado localmente e na nuvem. Ele suporta consultas de dados onde eles residem, incluindo Hive, Cassandra, bancos de dados relacionais e até mesmo datastores proprietários. Para mecanismos de autenticação, ele suporta autenticação baseada em senha, LDAP e OAuth. Para autorização e controle de acesso, Trino fornece a capacidade de conceder acesso nos níveis de catálogo, esquema e tabela. Nossas equipes usaram grupos de recursos divididos de acordo com padrões de consumo, como casos de uso de visualização, relatórios ou aprendizado de máquina, para gerenciar e limitar o uso de recursos. O monitoramento baseado em JMX fornece um conjunto de métricas rico para permitir a atribuição de custos no nível da consulta ou da pessoa usuária. Nossas equipes usam Trino como uma porta de entrada para o acesso a dados em uma variedade de fontes. Quando se trata de uma consulta de dados em escala extremamente grande, Trino é uma aposta segura para nossas equipes. Presto, o projeto do Facebook do qual Trino origina, foi destaque no Radar em novembro de 2015.

32. Wiz

Experimente

Wiz é um dos concorrentes no ecossistema de plataformas de segurança de nuvem que permite às pessoas usuárias prevenir, detectar e responder a riscos e ameaças de segurança em uma única plataforma. Wiz pode detectar e alertar sobre configurações incorretas, vulnerabilidades e segredos vazados em artefatos que ainda não foram implantados em ambientes em execução (imagens de contêiner, código de infraestrutura) e também em workloads em execução (contêineres, VMs e serviços de nuvem). Ele também contextualiza descobertas no cenário de nuvem específico da cliente para permitir que as equipes de resposta entendam melhor o problema e priorizem as mitigações. Nossas equipes tiveram uma boa experiência com Wiz. Acreditam que Wiz está evoluindo rapidamente e adicionando novos recursos, e valorizam que a plataforma permite a detecção de riscos e ameaças antes de algumas outras ferramentas semelhantes, pois analisa alterações de forma contínua.

33. ActivityPub

Avalie

Com a atual movimentação no ecossistema de plataformas de microblog, o protocolo **ActivityPub** está ganhando destaque. ActivityPub é um protocolo aberto para compartilhar informações como postagens, publicações e datas. Ele pode ser usado para implementar uma plataforma de mídia social, mas o principal benefício é que ele oferece interoperabilidade entre diferentes plataformas de mídia social. Esperamos que o ActivityPub desempenhe um papel significativo nesse espaço, mas o mencionamos aqui porque estamos curiosas com as possibilidades para além dos casos de uso óbvios nas mídias sociais. Um exemplo é o suporte do ActivityPub para requisições de merge, recentemente proposto para o GitLab.

34. Aplicativos de Contêiner do Azure

Avalie

Os aplicativos de contêiner do Azure é um namespace Kubernetes gerenciado como serviço que simplifica a implantação de cargas de trabalho em contêineres, eliminando a necessidade de manutenção complexa de clusters Kubernetes e componentes de infraestrutura subjacentes, reduzindo assim os encargos operacionais e administrativos. No entanto, é importante ter cuidado ao considerar essa opção; atualmente em fase de desenvolvimento, ele tem apresentado inconsistências na representação de suas capacidades no portal Azure e encontra obstáculos de integração, particularmente com o provedor Terraform padrão para Azure que não reflete de forma satisfatória as funcionalidades reais da ferramenta. Dado tudo isso, recomendamos avaliar esta ferramenta com cuidado.

35. Serviço OpenAI do Azure

Avalie

Com o grande interesse em IA generativa, muitas soluções surgiram para acessar os principais modelos. Se você está considerando ou já usa Azure, vale a pena avaliar o **Serviço OpenAI do Azure**. Ele fornece acesso aos modelos GPT-4, GPT-35-Turbo e Embeddings da OpenAI por meio de uma API REST, um SDK Python e uma interface web. Os modelos podem ser adaptados para tarefas, como geração de conteúdo, resumo, busca semântica e tradução de linguagem natural para código. O ajuste fino também está disponível por meio de aprendizado de poucos exemplos e personalização de hiperparâmetros. Em comparação com a API própria da OpenAI, Serviço OpenAI do Azure se beneficia dos recursos de segurança e conformidade de nível empresarial do Azure; também está disponível em mais regiões, embora a disponibilidade seja limitada para cada uma das maiores regiões geográficas, e, até o momento, a Índia não está incluída.

36. ChatGLM

Avalie

Existem muitos modelos de linguagem de grande porte (LLMs) emergentes no mundo de língua inglesa. Embora esses modelos sejam geralmente pré-treinados em vários idiomas, seu desempenho em outros idiomas pode não ser tão bom quanto em inglês. O **ChatGLM**, desenvolvido pela Universidade Tsinghua, é um modelo de linguagem bilíngue aberto e otimizado para conversação em chinês com base na arquitetura do General Language Model. Como o chinês pode ser mais complexo do que o inglês, com sua diferente segmentação de palavras e gramática, é importante ter um LLM otimizado para o chinês. Nossa equipe descobriu que o ChatGLM superou outros LLMs em precisão e robustez quando construímos um aplicativo de detecção de emoções em chinês para um call center. Considerando que muitos LLMs não estão disponíveis na China devido a restrições de licenciamento ou regionais, o ChatGLM tornou-se uma das poucas opções de código aberto.

37. Chroma

Avalie

Chroma é um repositório de armazenamento de vetores e banco de dados de embeddings de código aberto, útil para aprimorar aplicativos movidos por modelos de linguagem de grande porte (LLMs) ao facilitar o armazenamento e utilização de conhecimento de domínio em LLMs, que normalmente não possuem memória interna. Particularmente, em aplicações de texto para texto, Chroma pode automatizar o processo intrincado de geração de embeddings de palavras e análise de similaridades entre eles e embeddings de consulta, o que simplifica consideravelmente as operações. Ele também oferece a opção de armazenar embeddings personalizados, promovendo uma combinação de automação e personalização. Dadas suas capacidades de aprimorar a funcionalidade de aplicativos movidos a LLM, aconselhamos as equipes a avaliar o Chroma, aproveitando seu potencial para refinar a maneira como o conhecimento de domínio é integrado a esses aplicativos.

38. Kraftful

Avalie

Plataformas de pesquisa de UX como o Dovetail oferecem às organizações uma ferramenta para entender e melhorar a experiência de suas clientes. Com ela, as empresas podem obter de forma rápida e fácil insights valiosos sobre as necessidades, preferências e comportamentos de clientes, coletando e analisando dados de feedback da cliente, pesquisas, entrevistas e muito mais. A análise de sentimento, segmentação de clientes, pesquisa de mercado ou análise de dados e geração de insights são tarefas importantes no desenvolvimento de produtos - essas tarefas combinam com o que os LLMs (modelos de linguagem de grande porte) são bons, por isso vemos um grande potencial de disrupção no campo de desenvolvimento de produtos.

Kraftful — que se auto-declara o Copilot para construtores de produtos — saiu na frente. Ainda está em beta e você deve fornecer seu e-mail até para acessar a lista de funcionalidades disponíveis. Nós brincamos com ele e vimos ótimos resultados. Você pode conectar mais de 30 fontes de feedback de usuário à plataforma e ela analisará os dados e identificará solicitações de funcionalidades, reclamações comuns, o que as pessoas usuárias adoram no produto e até mesmo nomeará seus concorrentes. Para obter mais detalhes, você pode fazer perguntas como faria ao ChatGPT ou Google Bard — o benefício aqui é que ele é otimizado para seus dados. Depois de priorizar o que será feito a partir do feedback de usuário, o Kraftful gera user stories para você com base em todos os dados subjacentes, incluindo critérios de aceitação, tornando-o um grande assistente para até mesmo gerentes de produto e analistas de negócios muito experientes.

39. pgvector

Avalie

Com o aumento de aplicações de IA generativa, observamos um padrão de armazenamento e busca eficiente de vetores de embeddings por similaridades. **pgvector** é uma extensão de código aberto para PostgreSQL que permite a busca de similaridade de vetores. Nós gostamos dele porque nos permite buscar os embeddings no PostgreSQL sem mover os dados para outro local apenas para a busca de similaridade. Embora existam vários motores de busca de vetores especializados, queremos que você avalie pgvector.

40. Pinecone

Avalie

Pinecone é um banco de dados de vetores completamente gerenciado, acessível às pessoas desenvolvedoras, e nativo da nuvem com uma API simples e sem complicações de infraestrutura. Pinecone fornece resultados de consulta filtrados com baixa latência na escala de bilhões de vetores. Nossas equipes constataram que os bancos de dados de fornecedores, e o Pinecone em particular, são muito úteis e rápidos de começar a usar em casos de uso como armazenar a base de conhecimento de uma equipe, ou o conteúdo do portal de suporte ao cliente, em vez de fazer ajuste fino em LLMs complexos.

41. wazero

Avalie

wazero é um runtime WebAssembly (WASM) sem dependências escrito em Go. Embora o runtime em si seja neutro em relação à linguagem, queremos destacar o wazero para as pessoas desenvolvedoras Go, porque oferece uma maneira adequada de extensão dos programas Go, com módulos WASM escritos em qualquer linguagem em conformidade. Não há uma dependência de CGO, por isso, você pode compilar seus aplicativos Go com mais facilidade para outras plataformas. Embora você tenha opções quando se trata de tempos de execução para WASM, achamos que o wazero vale a pena ser avaliado.

Ferramentas

Adote

42. dbt
43. Mermaid
44. Ruff
45. Snyk

Experimente

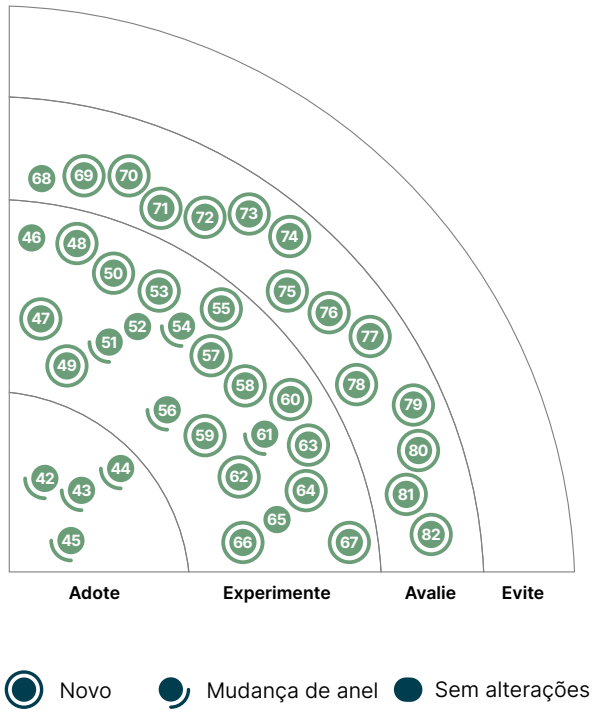
46. AWS Control Tower
47. Bloc
48. cdk-nag
49. Checkov
50. Chromatic
51. Cilium
52. Cloud Carbon Footprint
53. Container Structure Tests
54. Devbox
55. DX DevEx 360
56. GitHub Copilot
57. Insomnia
58. Plugin HTTP Client do IntelliJ
59. KEDA
60. Kubeconform
61. mob
62. MobSF
63. Mocks Server
64. Prisma runtime defense
65. Terratest
66. Thanos
67. Yalc

Avalie

68. ChatGPT
69. Codeium
70. Fila de merges do GitHub
71. Google Bard
72. Google Cloud Workstations
73. Gradio
74. KWOK
75. Llama 2
76. Maestro
77. Modelos de linguagem de grande porte (LLM) de código aberto para programação
78. OpenCost
79. OpenRewrite
80. OrbStack
81. Pixie
82. Tabnine

Evite

—



42. dbt

adote

dbt continua sendo nossa ferramenta preferida para a transformação de dados no fluxo ELT. Nós gostamos que a ferramenta se adequa ao rigor de engenharia e habilita práticas como modularidade, testabilidade e reusabilidade de transformações baseadas em SQL. dbt está disponível em versões de código aberto e SaaS comercial e tem um ecossistema saudável, incluindo uma central para a comunidade com pacotes para teste unitário, qualidade de dados e observabilidade dos dados, para mencionar alguns. Alguns dos pacotes de extensão que queremos destacar incluem: dbt-expectations e dbt-unit-testing que facilitam armazenamento de dados em nuvem, lakehouses e bancos de dados incluindo, Snowflake, BigQuery, Redshift, Databricks e Postgres. Quando estamos trabalhando com dados estruturados onde é possível configurar a transformação dos dados em SQL, nosso time prefere dbt - e é por isso que estamos movendo-o para Adote.

43. Mermaid

Adote

Mermaid permite gerar diagramas a partir de uma linguagem de marcação semelhante ao Markdown. Desde que o apresentamos no Radar, Mermaid adicionou suporte para mais diagramas e integrações com repositórios de código, IDEs e ferramentas de gerenciamento de conhecimento. De maneira notável, é adotado de forma nativa em repositórios de código populares como GitHub e GitLab, permitindo a fácil incorporação e atualização de diagramas Mermaid no meio da documentação Markdown. Muitas de nossas equipes preferem usar o Mermaid como sua ferramenta de diagrama-como-código devido à sua facilidade de uso, diversidade de integrações e ampla variedade de tipos de diagramas suportados, que continuam crescendo.

44. Ruff

Adote

Ruff é um linter relativamente novo para Python. Quando se trata de linters, para nós a questão não é sobre usar, mas qual devemos usar. Ruff se destaca por sua experiência pronta para uso e sua velocidade. São mais de 500 regras internas, o que substitui facilmente o Flake8, incluindo muitos dos plugins dele. As afirmações feitas pelo time do Ruff sobre seu desempenho são confirmadas pela nossa experiência de uso; é realmente mais rápido que outros linters, em pelo menos uma ordem de magnitude, o que é um benefício enorme, pois ajuda a reduzir o tempo de compilação em grandes bases de código. Por esses motivos, Ruff se tornou nossa escolha padrão para linters Python.

45. Snyk

Adote

Snyk oferece testes de segurança de aplicações estáticas (SAST) e análise de componentes de software (SCA) para ajudar você a encontrar, corrigir e monitorar problemas de segurança ao longo do ciclo de vida de desenvolvimento de software. Sua ampla gama de recursos é projetada para acelerar o ciclo de feedback, favorecendo segurança na fase inicial em vez do antipadrão sanduíche de segurança. Como uma das melhores plataformas de segurança disponíveis hoje, Snyk se destaca por sua capacidade de identificar uma gama mais ampla de problemas, habilitada principalmente por uma equipe de pesquisa dedicada adicionando à sua base de dados de vulnerabilidades. Mas há espaço para melhorias: o painel atualmente não oferece uma maneira fácil de filtrar o ruído para informações específicas e acionáveis; dependendo do ecossistema de linguagem, as integrações baseadas em SCA podem gerar falsos positivos em comparação com as integrações baseadas em pipeline porque o Snyk precisa adivinhar quais são as dependências resolvidas; a resolução automatizada não é consistentemente bem-sucedida; e é necessário um investimento significativo em integração para alcançar um controle de acesso adequado ou estabelecer um SBOM em ambientes de alta regulamentação. Apesar dessas limitações, muitos de nossos clientes corporativos adotaram o Snyk; e nós também o usamos em nossa área de TI interna.

46. AWS Control Tower

Experimente

O gerenciamento de multicontas é um desafio na AWS, especialmente na configuração e governança. **AWS Control Tower** aborda esse desafio simplificando a configuração e automatizando a governança; ele atende aos requisitos regulatórios com diretrizes. O AWS Control Tower possui uma Fábrica de Contas (Account Factory) integrada que ajuda a automatizar o workflow de provisionamento de contas. Entre outras coisas, você pode atualizar, desassociar e fechar contas criadas e provisionadas por você por meio da Fábrica de Contas. Devido à sua falta de automação e personalização, a Amazon introduziu o AWS Control Tower Account Factory for Terraform (AFT). O AFT permite que você provisione personalizações para enviar webhooks ou tomar ações específicas que permitem a integração com outras ferramentas para tarefas de partida como parte do processo de criação da conta. Um dos casos de uso aproveitados por nossa equipe foi gerenciar um conjunto de itens pré-configurados para contas que tinham configurações de "defina e esqueça" (set-and-forget) para referência e criação de acesso para roles do GitHub Actions. Isso resultou em fornecer às pessoas desenvolvedoras uma conta com segurança padrão estabelecida com um VPC totalmente integrado, pronto para receber workload via GitHub Actions. Nossas equipes relataram ótimos resultados usando AWS Control Tower para gerenciar contas, como um único controle de acesso para várias equipes, e com o uso do AFT em suas tarefas.

47. Bloc

Experimente

Bloc é uma biblioteca de gerenciamento de estado reativo para Flutter. Dentre as opções de gerenciamento de estado disponíveis para Flutter, queremos destacar Bloc porque nossas equipes tiveram uma boa experiência com a biblioteca ao construir aplicativos móveis complexos. A organização estrutural do código em torno do padrão BLoC resultou em uma nítida separação da lógica de negócios da camada de apresentação, pois os Widgets da IU comunicam-se com a lógica de negócios por meio de streams e sinks de eventos. Bloc também tem um bom suporte a plugins em ambas IDEs IntelliJ e VSCode.

48. cdk-nag

Experimente

O **cdk-nag** identifica e relata problemas de segurança e conformidade em aplicações AWS CDK ou modelos do CloudFormation. Ele vem com vários pacotes de regras: um pacote geral da AWS que inclui verificações para o que a AWS considera as melhores práticas, bem como pacotes para conformidade com HIPAA, NIST e PCI. Você pode adicionar regras conforme necessário. As regras podem resultar em avisos ou erros, ambos incluídos nos relatórios gerados pela ferramenta. Quando há erros, o comando cdk deploy não fará implantações. Se a causa do erro não puder ser corrigida a tempo, você ainda pode implantar com o erro presente, mas suprimido. Obviamente, isso só deve ser feito em casos excepcionais.

49. Checkov

Experimente

Checkov é um escâner de segurança estático especializado em infraestrutura como código (IaC). Ele suporta uma ampla gama de linguagens de infraestrutura, incluindo manifestos do Kubernetes, gráficos do Helm, modelos do CloudFormation e Terraform. Facilmente implantável em pipelines de CI/CD, ele protege contra potenciais lacunas de segurança em diversas configurações de infraestrutura de nuvem. Aproveitando-se de um conjunto de regras padrão, ele identifica cenários de segurança comuns com conselhos de remediação detalhados disponíveis em seu site. Checkov suporta regras personalizadas e usa YAML para definições de diretrizes simples ou Python para criar regras mais complexas. Nossas equipes usaram com sucesso o Checkov para aprimorar a segurança durante as implantações de infraestrutura, apreciando os avisos adiantados que ele fornece sobre potenciais problemas antes da implantação.

50. Chromatic

Experimente

Chromatic é uma ferramenta de teste de regressão visual para ajudar a detectar regressões de IU em aplicações web. Ele funciona fazendo snapshots de componentes de IU e comparando-os com snapshots anteriores quando eles são alterados. É um serviço hospedado que se integra com serviços populares de hospedagem de código em nuvem. Construído sobre o Storybook, ele faz teste de regressão visual de componente. Ele pode renderizar os componentes em diferentes tamanhos de tela para testes responsivos e integra-se com fluxos de trabalho de CI, gerando o changeset da IU para cada commit, o que facilita a revisão. Nossas equipes acham que a diferença visual entre Chromatic e outras ferramentas neste espaço é muito melhor; e a capacidade de destacar as alterações visualmente o torna particularmente útil.

51. Cilium

Experimente

eBPF é famoso por ser uma aplicação transparente, de alto desempenho e baixa sobrecarga. Por isso, a comunidade tem explorado seu uso para malha de serviço sem sidecar. **Cilium** é um projeto de código aberto que fornece rede, segurança e observabilidade para ambientes nativos de nuvem, como clusters Kubernetes e outras plataformas de orquestração de contêineres. Ele fornece uma rede Layer 3 plana simples para roteamento ou sobreposição e é compatível com o protocolo L7. Ao desacoplar a segurança do endereçamento, Cilium pode desempenhar um papel significativo como uma nova camada de proteção de rede. Vimos a adoção do Cilium por alguns provedores de nuvem e também o usamos em projetos da Thoughtworks. A comunidade ainda está discutindo se o eBPF pode substituir o sidecar, mas parece haver consenso de que alguns recursos da malha (mesh) não podem ou não devem ser executados no kernel. Além disso, a aplicação do Cilium também requer experiência relacionada ao eBPF. Com base nos resultados positivos em nosso projeto, recomendamos que você experimente essa tecnologia.

52. Cloud Carbon Footprint

Experimente

Cloud Carbon Footprint (CCF) é uma ferramenta de código aberto que estima emissões de carbono para workloads na nuvem dos principais provedores de serviços de nuvem. Ele consulta as APIs da nuvem para obter dados de uso de recursos e usa várias fontes para rastrear emissões de carbono. Seguindo uma metodologia publicada, o CCF combina essas informações em estimativas de emissões e fornece uma visualização dos dados ao longo do tempo. Os provedores de nuvem começaram a adicionar ofertas semelhantes às suas plataformas, mas as organizações ainda estão implantando o CCF porque ele possui todos os recursos: é de código aberto, foi projetado para ser expandido, funciona em várias nuvens e tem uma metodologia transparente e publicada. Além disso, ele também inclui estimativas para emissões de escopo 2 e escopo 3 — para uso de eletricidade e produção de hardware, respectivamente. Em nossos experimentos, as estimativas entre diferentes ferramentas variaram, o que não é uma grande surpresa, dado que todas as ferramentas nessa área fazem estimativas e multiplicam números estimados. No entanto, definir uma ferramenta, estabelecer uma linha de base e melhorar a partir dessa linha de base é o cenário de uso principal que encontramos, e ferramentas como Kepler podem reduzir a necessidade de estimativas no futuro. O CCF também fornece recomendações de otimização para GCP e AWS, que não só ajudam a reduzir sua pegada de carbono na nuvem, mas também podem se tornar parte de uma estratégia mais ampla de otimização de custos de nuvem. A Thoughtworks é uma importante contribuidora do CCF.

53. Container Structure Tests

Experimente

Container Structure Tests (CST) é uma ferramenta desenvolvida pelo Google para testar a estrutura de uma imagem de contêiner. O CST pode ser usado para verificar a existência ou ausência de um determinado arquivo no sistema de arquivos da imagem, para verificar o conteúdo de um arquivo, para verificar a saída ou erros de um comando específico executado no contêiner e para verificar os metadados da imagem do contêiner (ou seja, labels, entrypoint e command), o que ajuda a garantir a conformidade com o CIS Docker Benchmark. Temos tido boas experiências com o CST e recomendamos que você experimente. Além de prevenir vulnerabilidades — verificando se o contêiner está expondo portas desnecessárias — também o usamos para validar que cada contêiner Docker passa por todos os requisitos necessários para ser implantado e para executar um aplicativo na plataforma da empresa. Um desses requisitos era ter um agente de observabilidade instalado na imagem. É importante estar ciente de que o CST não é oficialmente suportado pelo Google, o que pode afetar sua manutenção.

54. Devbox

Avalie

Devbox é uma ferramenta de linha de comando que fornece uma interface de fácil acesso para criar ambientes de desenvolvimento reproduzíveis, por projeto, aproveitando o gerenciador de pacotes Nix sem usar máquinas virtuais ou contêineres. Nossas equipes o utilizam para eliminar incompatibilidades de versão e configuração de ferramentas CLI e scripts personalizados em seus ambientes de desenvolvimento por projeto, além da padronização que os gerenciadores de pacotes por linguagem fornecem. Elas descobriram que isso simplifica significativamente o workflow de onboarding porque, uma vez que ele foi configurado para um repositório de código, basta um comando CLI (devbox shell) para instalá-lo em uma nova máquina. O Devbox suporta hooks de shell, scripts personalizados e geração de devcontainer.json para integração com VSCode.

55. DX DevEx 360

Experimente

DX DevEx 360 é uma ferramenta baseada em questionários que ajuda a encontrar sinais precursores de produtividade das pessoas desenvolvedoras, concentrando-se nas fricções que elas enfrentam em sua rotina, como o processo de revisão de código, qualidade do código, capacidade de se concentrar em uma tarefa complexa sem interrupções e muito mais. O questionário foi desenvolvido por Nicole Forsgren e Margaret-Anne Storey, que lideraram esforços anteriores como DORA e SPACE, entre outras especialistas. Nossas equipes de engenharia de plataforma usaram DX DevEx 360 com sucesso para entender como as pessoas desenvolvedoras se sentiam e identificar pontos de fricção para informar o roadmap da plataforma. Ao contrário de ferramentas semelhantes, com DX DevEx 360, obtivemos uma taxa de resposta de 90% ou mais, muitas vezes com comentários detalhados das pessoas desenvolvedoras sobre problemas e ideias de melhorias. Também gostamos que a ferramenta torna os resultados transparentes para as pessoas engenheiras da empresa, e não apenas para a gestão, e que permite a análise por equipe e proporciona a melhoria contínua de acordo com o contexto do time.

56. GitHub Copilot

Experimente

GitHub Copilot é usado por muitas de nossas equipes para ajudá-las a escrever código mais rápido. De modo geral, a maioria de nossas pessoas desenvolvedoras acha a ferramenta muito útil e não gostariam de parar de usá-la. Estamos coletando e compartilhando muitas dessas nossas experiências com o Copilot por meio de uma série sobre IA Generativa e um guia de introdução ao uso de Copilot. É importante dizer que o GitHub Copilot pode ser usado com qualquer código,

não apenas com códigos hospedados no GitHub. Também ficamos animados ao perceber que o recurso de bate-papo do Copilot do roadmap do Copilot X teve sua disponibilidade ampliada desde que o apresentamos no Radar. É uma poderosa adição ao recurso de assistência in-line do Copilot. A disponibilidade de um chat dentro da IDE torna mais fácil a descoberta das informações mais procuradas e sua integração com o editor torna mais fácil explorar erros ou pedir ao chat para auxiliar com tarefas relacionadas ao código em foco.

57. Insomnia

Experimente

Desde que o Postman anunciou em maio de 2023 que iria descontinuar o modo Scratch Pad com suas capacidades offline, as equipes que precisam manter os dados do espaço de trabalho da API fora de servidores de terceiros tiveram que encontrar alternativas. **Insomnia** é uma delas: é um aplicativo desktop gratuito e de código aberto projetado para teste, desenvolvimento e depuração de APIs. Embora Insomnia suporte sincronização online, ele permite que você mantenha os dados do espaço de trabalho da API offline. Nossas equipes consideraram a migração do Postman para o teste manual de API sem problemas, pois seus recursos são semelhantes, e ele permite a importação de coleções do Postman. Apesar das experiências positivas de nossas equipes com Insomnia, estamos de olho em outras alternativas em desenvolvimento de várias formas - de ferramentas GUI similares ao Insomnia, que são alternativas drop-in, a ferramentas CLI como o HTTPie, e plugins de IDE como o plugin HTTP Cliente do IntelliJ.

58. Plugin HTTP Client do IntelliJ

Experimente

O **plugin HTTP Client do IntelliJ** permite que pessoas desenvolvedoras criem, editem e executem requisições HTTP no editor de código, simplificando o processo de desenvolvimento para criar e consumir APIs. Ele está se tornando cada vez mais popular entre nossas equipes, que apreciam que suas funcionalidades são amigáveis às pessoas usuárias e sua conveniência. Suas funcionalidades mais notáveis incluem suporte a arquivos privados, que protegem chaves sensíveis ao excluí-las do git por padrão, controle de versão e a capacidade de utilizar variáveis, o que aprimora a experiência da pessoa desenvolvedora. Dada sua capacidade de otimizar os fluxos de trabalho das pessoas desenvolvedoras e reforçar as medidas de segurança, recomendamos experimentar esta ferramenta.

59. KEDA

Experimente

KEDA, o redimensionador automático (Autoscaller) de Eventos Kubernetes, faz exatamente o que o nome sugere: ele permite a escalabilidade de um cluster Kubernetes em relação ao número de eventos que devem ser processados. Em nossa experiência, é preferível usar indicadores antecessores (leading indicators) como a profundidade da fila (queue depth) - em vez de indicadores defasados como o uso da CPU. KEDA suporta diferentes fontes de eventos e vem com um catálogo de mais de 50 redimensionadores para várias plataformas de nuvem, bancos de dados, sistemas de mensagens, sistemas de telemetria, sistemas CI/CD e muito mais. Nossas equipes relatam que a facilidade com que o KEDA pode ser integrado permitiu manter a funcionalidade em microsserviços no Kubernetes, onde, de outra forma, eles poderiam ter considerado migrar parte do código de tratamento de eventos para funções sem servidor.

60. Kubeconform

Experimente

Kubeconform é uma ferramenta simplificada para validar manifestos e definições de recursos personalizados (CRD) do Kubernetes. Facilmente implantável em pipelines CI/CD ou máquinas locais, ele promove confiança ao validar recursos antes do deploy, mitigando potenciais erros. Dado seu histórico de aprimoramento da garantia operacional, especialmente com recursos modelos compartilhados entre equipes, recomendamos experimentar Kubeconform para reforçar a segurança e eficiência de seus processos de validação de recursos.

61. mob

Experimente

mob é uma ferramenta de linha de comando para um git handover sem interrupções para programação em par ou em grupo remota. Ele esconde todos os elementos de controle de versão atrás de uma interface de linha de comando, o que torna a participação em sessões de programação em grupo mais simples. Também oferece conselhos específicos sobre como participar remotamente, por exemplo, "pegar a tela" no Zoom em vez de encerrar uma tela compartilhada, garantindo que o layout do vídeo não seja alterado para as participantes. Várias de nossas equipes recomendam mob, e ele se tornou parte integrante do nosso conjunto de ferramentas em programação em par ou em grupo remota.

62. MobSF

Experimente

MobSF é uma ferramenta de código aberto, para automação de teste de segurança estática e dinâmica para detectar vulnerabilidades de segurança em aplicativos móveis iOS e Android. Ele escaneia tanto as fontes dos aplicativos quanto os binários e fornece relatórios detalhados sobre as vulnerabilidades. MobSF é distribuído como imagens Docker e vem com APIs REST fáceis de usar e, via mobsfscan, pode ser integrado em pipelines CI/CD. Nossa experiência com o uso do MobSF para teste de segurança de aplicativos Android tem sido positiva; recomendamos experimentá-lo para suas necessidades de teste de segurança de aplicativos móveis.

63. Mocks Server

Experimente

Mocks Server é uma ferramenta de mocking de API baseada em Node.js que é muito valorizada por nossas equipes pela sua capacidade de replicar respostas, cabeçalhos e códigos de status complexos de API. A geração de resposta dinâmica suporta a simulação de diversos cenários, o que permite o teste rigoroso de interações de API. Os mocks podem ser descritos como YAML ou JSON e gerenciados por meio de CLI, API REST ou código JavaScript. Os recursos do Mocks Server incluem correspondência de solicitações, proxy e recursos de reprodução de registro, que facilitam a emulação de interações realistas com a API. Gostamos especificamente da integração com contêineres Docker, o que torna fácil implantar o servidor de forma consistente entre os ambientes para que ele possa ser versionado e mantido como outro artefato do ecossistema. Sua abordagem direta se alinha com nosso foco na simplicidade e eficiência nos processos de desenvolvimento. Esperamos usar o Mocks Server mais extensivamente à medida que nossa estratégia de teste evolui junto com nossas soluções.

64. Prisma runtime defense

Experimente

Prisma runtime defense, que faz parte do conjunto de produtos Prisma Cloud, oferece uma nova abordagem à segurança de contêineres. Ele emprega um mecanismo para construir um modelo do comportamento esperado de um contêiner e, em seguida, detecta e bloqueia atividades anômalas quando alguma variação é encontrada em tempo de execução. Ele monitora processos de contêineres, atividades de rede e sistemas de arquivos em busca de padrões e alterações que indiquem que um ataque pode estar em andamento e bloqueia de acordo com as regras configuradas. Os modelos que aprendem o que constitui comportamentos "normais" são construídos a partir da análise estática de imagens Docker e da análise comportamental dinâmica por um período pré-configurado. Nossas equipes consideraram os resultados de nosso uso promissores.

65. Terratest

Experimente

Terratest continua sendo uma opção interessante para testes de infraestrutura. É uma biblioteca Golang que facilita a escrita de testes automatizados. Usando ferramentas de infraestrutura como código, como o Terraform, você pode criar componentes de infraestrutura real (como servidores, firewalls ou balanceadores de carga) para implantar aplicativos e depois validar o comportamento esperado usando o Terratest. No final do teste, o Terratest pode retirar os aplicativos e limpar os recursos. Nossas equipes relatam que essa abordagem para teste dos componentes de infraestrutura implantados promove a credibilidade da infraestrutura como código. Observamos nossas equipes escrevendo uma variedade de testes de segurança de infraestrutura para componentes de aplicativos e suas integrações. Isso inclui detectar configurações incorretas, verificar o controle de acesso (por exemplo, para garantir se certos papéis ou permissões IAM estão configurados corretamente ou para garantir que apenas usuárias autorizadas tenham acesso a recursos específicos) e realizar testes de segurança de rede para validar a prevenção de tráfego não autorizado para recursos sensíveis, entre outros. Isso permite que os testes de segurança aconteçam na fase inicial e forneçam feedback durante o próprio desenvolvimento.

66. Thanos

Experimente

Embora Prometheus continue sendo uma opção robusta para um stack de observabilidade autogerenciada, muitas equipes que gerenciam sistemas distribuídos modernos nativos de nuvem esbarram em suas limitações de nó único à medida que suas métricas crescem em quantidade e volume, e quando começam a precisar de uma configuração de alta disponibilidade. **Thanos** estende Prometheus adicionando recursos que o tornam adequado para monitoramento em grande escala, de longo prazo e altamente disponível. Ele faz isso, por exemplo, introduzindo componentes que irão ler os dados de instâncias do Prometheus e os armazenar em repositórios de objetos, gerenciar a retenção e compactação nos repositórios de objetos e federar as consultas em várias instâncias do Prometheus. Nossas equipes consideraram a migração do Prometheus sem problemas significativos, pois o Thanos mantém a compatibilidade com a API de consulta do Prometheus. Isso significa que eles podem continuar usando seus painéis existentes, ferramentas de alerta e outras ferramentas que integram à API do Prometheus. Embora nossas equipes tenham sido bem-sucedidas com Thanos, também recomendamos ficar de olho no Cortex, como uma outra forma de estender o Prometheus.

67. Yalc

Experimente

Yalc é um repositório simples de pacotes JavaScript e uma ferramenta para publicar e usar pacotes em um ambiente de desenvolvimento local. É uma alternativa mais confiável ao comando npm link que tem algumas restrições. Yalc é útil quando se trabalha com vários pacotes, especialmente quando alguns usam yarn e outros usam npm. Também é útil para testar os pacotes localmente antes de publicá-los em um registry remoto. Em nossa experiência, o Yalc é importante em uma configuração de múltiplos pacotes e acelera front-end e outros fluxos de trabalho de desenvolvimento de aplicações JavaScript.

68. ChatGPT

Avalie

ChatGPT continua a atrair atenção. Os casos de uso imaginativos e abordagens inovadoras de prompting significam que ele está ganhando utilidade crescente ao longo do tempo. GPT4, o modelo de linguagem de grande porte (LLM) que alimenta o ChatGPT, agora também tem a capacidade de se integrar a ferramentas externas, como um repositório de gerenciamento de conhecimento, ambiente de programação isolado (sandboxed) ou pesquisa na web. A recente introdução do ChatGPT Enterprise pode ajudar a aliviar as preocupações de propriedade intelectual, ao mesmo tempo que fornece recursos "corporativos" como rastreamento de uso e melhor gerenciamento de usuárias por meio de SSO.

Embora a capacidade do ChatGPT de "escrever" código tenha sido muito louvada, acreditamos que as organizações devam utilizá-lo em todo o ciclo de vida do software para melhorar a eficiência e reduzir erros. Por exemplo, o ChatGPT pode fornecer perspectivas ou sugestões adicionais para tarefas tão diversas quanto análise de requisitos, design arquitetural ou engenharia reversa de sistemas legados. Ainda acreditamos que o ChatGPT é melhor utilizado como pontapé inicial em um processo - como ajudar com um primeiro rascunho de uma user story ou o modelo para uma tarefa de programação - em vez de uma ferramenta que produz resultados "totalmente prontos". Dito isso, suas capacidades estão melhorando a cada semana, e algumas tarefas de programação podem agora ser totalmente possíveis por meio de um prompt cuidadoso, que é uma arte por si só.

69. Codeium

Avalie

No ecossistema de assistentes de programação com IA, **Codeium** é um dos produtos mais promissores. Semelhante ao Tabnine, Codeium busca endereçar algumas das maiores preocupações das empresas sobre o uso de assistentes de programação: ele reduz alguns dos receios de licenciamento de código aberto, porque não treina seus modelos com código de repositórios com licenças não permissivas. Também oferece auto-hospedagem para a ferramenta, assim não há necessidade de enviar seus fragmentos de código para um serviço de terceiros. O Codeium se destaca por seu amplo suporte a IDEs e serviços de notebook, e embora não esteja disponível há tanto tempo quanto o GitHub Copilot ou o Tabnine, nossas primeiras impressões do produto foram positivas.

70. Fila de merges do GitHub

Avalie

Há muito tempo defendemos o uso de branches de desenvolvimento de curta duração que são mesclados (merged) frequentemente no branch principal, que é então mantido pronto para implantação. Essa prática de desenvolvimento (trunk-based) está de mãos dadas com a integração contínua e, quando as condições permitem, resulta em ciclos de feedback mais rápidos e em um fluxo de desenvolvimento mais eficiente. No entanto, nem todos preferem essa abordagem, e frequentemente, adaptamos nosso estilo para acomodar as práticas das nossas clientes. Às vezes, isso inclui branches de longa duração seguidos de pull requests que devem ser revisadas e aprovadas

manualmente antes de serem mescladas no branch principal. Nessas situações, usamos o novo recurso de **fila de merges do GitHub**. Ele permite que os pull requests recebidos sejam automaticamente colocados em fila e mesclados (merged) em um branch especial na ordem em que foram recebidos. Em seguida, temos a opção de automatizar nossas próprias "verificações de merge" para evitar commits incompatíveis. Isso essencialmente simula o desenvolvimento direto no tronco (trunk-based) (mesmo que os Pull Requests ainda não tenham sido mesclados no branch principal do código) e permite que as pessoas desenvolvedoras testem suas funcionalidades no contexto sem ter que esperar que o Pull Request seja aprovado. Com a fila de merges do GitHub, você obtém os benefícios do desenvolvimento direto no tronco (trunk-based) mesmo quando não pode se comprometer totalmente com ele.

71. Google Bard

Avalie

Google Bard é um chatbot de IA generativa desenvolvido pelo Google AI. Assim como o ChatGPT, ele é conversacional e capaz de se comunicar e gerar texto semelhante ao humano em resposta a uma ampla gama de prompts e perguntas. O Bard é alimentado pelo Pathways Language Model (PaLM 2) do Google, que é um modelo de linguagem de grande porte (LLM) treinado em um grande conjunto de dados de texto e código. O Bard é capaz de gerar texto, traduzir idiomas, escrever diferentes tipos de conteúdo criativo e responder às suas perguntas de forma informativa. O Bard também pode ser usado como uma ferramenta de orientação para o desenvolvimento de software. Às vezes, nossas pessoas desenvolvedoras o acham útil para obter algumas sugestões de programação ou melhores práticas e configurações de infraestrutura para diferentes cenários. Também experimentamos o Bard para as traduções de idiomas do Radar e obtivemos bons resultados para criar o primeiro rascunho do texto. Embora a ferramenta ainda esteja em desenvolvimento, razão pela qual pode ser um pouco mais lenta em comparação com o ChatGPT, encorajamos as pessoas desenvolvedoras a explorar a ferramenta e avaliar seus potenciais benefícios.

72. Google Cloud Workstations

Avalie

Google Cloud Workstations é uma oferta de Ambiente de Desenvolvimento em Nuvem (Cloud Development Environment - CDE, em inglês) do Google Cloud Platform. Oferece ambientes de desenvolvimento totalmente gerenciados e conteinerizados que são acessíveis por meio de SSH, HTTPS, VSCode e IDEs Jetbrains, entre outros, dando às pessoas desenvolvedoras a ilusão de se conectar a um ambiente local. O Google Cloud Workstations permite que as pessoas administradoras tornem os ambientes de desenvolvimento conteinerizados parte de uma rede privada e que sejam acessíveis publicamente ou privadamente. Essa capacidade de ajustar as configurações de rede, juntamente com o suporte para criar os ambientes com imagens personalizadas ou predefinidas, torna o Google Cloud Workstations, em nossa opinião, digno de ser avaliado por organizações que procuram uma solução CDE segura dentro de seu próprio perímetro GCP. Se você está considerando o Google Cloud Workstations, recomendamos que você teste sua configuração de rede antes de implantá-lo amplamente, pois a alta latência pode se tornar um grande obstáculo à experiência das pessoas desenvolvedoras com esses contêineres.

73. Gradio

Avalie

Gradio é uma biblioteca Python de código aberto que permite a criação rápida e fácil de interfaces web interativas para modelos de ML. Uma interface gráfica de usuário (GUI) sobre modelos de aprendizagem de máquina (ML) permite melhor compreensão das entradas, restrições e saídas por públicos não técnicos. Gradio suporta muitos tipos de entrada e saída - de texto e imagens a voz - e se tornou uma ferramenta essencial para prototipagem rápida e avaliação de modelos. Gradio

permite que você hospede suas demonstrações (demos) facilmente no Hugging Face ou execute-as localmente e permita que outras pessoas acessem a demo remotamente com uma url "XXXXX.gradio.app". Por exemplo, o famoso experimento DALL-E mini usa Gradio e é hospedado no Hugging Face Spaces. Nossas equipes ficaram satisfeitas em usar essa biblioteca para experimentação e prototipagem, razão pela qual a colocamos no anel Avalie.

74. KWOK

Avalie

KWOK (Kubernetes WithOut Kubelet) é uma ferramenta que simula o ciclo de vida de nós e pods falsos para testar o plano de controle de um cluster Kubernetes. É difícil realizar testes de estresse em controladores e operadores Kubernetes personalizados sem um cluster significativamente grande. No entanto, com KWOK, você pode facilmente configurar um cluster com milhares de nós em seu laptop sem consumir uma quantidade significativa de CPU ou memória. Essa simulação permite diferentes configurações de tipos de nós e pods para testar vários cenários e casos de borda. Se você precisa de um cluster real para testar seus operadores e definições de recursos personalizados (CRDs), recomendamos kind ou k3s; mas se você só precisa simular um grande número de nós falsos, sugerimos que você avalie KWOK.

75. Llama 2

Avalie

Llama 2, da Meta, é um modelo de linguagem de grande porte que é gratuito para uso tanto em pesquisa quanto comercial. Ele está disponível como um modelo pré-treinado bruto e, depois de ajustado, como Llama-2-chat para conversação e Code Llama para auto-completar código. Como está disponível em vários tamanhos — 7B, 13B e 70B — o Llama 2 é uma boa opção para um LLM auto-hospedado, se você quiser controlar seus dados. A Meta descreve o Llama 2 como "de código aberto", uma afirmação que tem atraído alguma crítica. A licença e a política de uso aceitável da Meta impõem restrições ao uso comercial para algumas pessoas usuárias e também restringem o uso do modelo e do software para determinados fins. Os dados de treinamento do Llama 2 não são abertos, o que pode dificultar a compreensão e a alteração do modelo. No entanto, a disponibilidade de um modelo poderoso e capaz, pelo menos de forma "semi-aberta", é bem-vinda.

76. Maestro

Avalie

Maestro é uma nova ferramenta para a automação de teste de interface de pessoas usuárias, móvel e multiplataforma, com tolerância integrada a instabilidades e variância nos tempos de carregamento do aplicativo, devido a fatores de rede ou externos. Com uma sintaxe YAML declarativa, ele facilita a escrita e a preservação de testes automatizados para aplicativos móveis. Ele suporta aplicativos nativos iOS e Android, React Native e Flutter, bem como uma variedade de recursos para automatizar interações complexas de IU móvel, como tocar, rolar e deslizar. O Maestro é distribuído como um único binário para facilitar o uso, é executado em modo interpretado e facilita a criação de novos testes graças a recursos como o modo contínuo. O Maestro ainda não possui recursos específicos, como suporte a dispositivos iOS, mas a ferramenta está evoluindo rapidamente.

77. Modelos de linguagem de grande porte (LLMs) de código aberto para programação

Avalie

GitHub Copilot é uma ferramenta valiosa para assistência de programação durante o desenvolvimento de software. Por baixo dos panos, os modelos de linguagem de grande porte (LLMs) podem oferecer ótimas experiências de desenvolvimento por meio de assistência de código em linha, ajuste fino de código, suporte conversacional na IDE e muito mais. A maioria desses modelos é proprietária e só pode ser usada por meio de serviços de assinatura. No entanto, existem vários **modelos de linguagem de grande porte (LLMs) de código aberto para programação** que podem ser usados. Se você precisar construir seu próprio serviço de assistência de programação (como para uma indústria altamente regulada), considere modelos como StarCoder e WizardCoder. StarCoder é treinado com um grande conjunto de dados mantido pelo BigCode. WizardCoder é um modelo StarCoder aprimorado com Evol-Instruct, um modelo de linguagem que pode gerar código de forma criativa. Usamos StarCoder em nossos experimentos e o consideramos útil para gerar elementos estruturados de engenharia de software, como código, YAML, SQL e JSON. Com base em nossos experimentos, descobrimos que ambos os modelos são receptivos à aprendizagem contextual usando exemplos de few-shot no prompt. No entanto, para tarefas subsequentes específicas (como geração de SQL para um banco de dados específico como Postgres), os modelos precisaram de ajuste fino. Recentemente, a Meta lançou o Code Llama, uma versão especializada para código do Llama 2. Recomendamos que você utilize com cautela esses modelos de código aberto. Considere suas licenças, as licenças dos códigos e dos conjuntos de dados usados para treinar os modelos. Avalie cuidadosamente esses aspectos antes de escolher qualquer um desses LLMs para sua organização.

78. OpenCost

Avalie

OpenCost é um projeto de código aberto para monitoramento de custos de infraestrutura que mostra os valores na granularidade de objetos Kubernetes (pods, containers, clusters, etc.), cobrindo vários recursos intra-cluster (CPU, GPU, RAM, armazenamento, rede). Ele tem integrações com várias APIs de provedores de nuvem para obter dados de faturamento e pode ser configurado com preços para clusters Kubernetes locais. O OpenCost é o motor de alocação de custos originalmente construído e ainda usado pelo Kubecost, mas também pode ser usado sozinho. Os dados de alocação de custos do OpenCost podem ser exportados para arquivos CSV ou para Prometheus para análise e visualização adicionais. Nossas equipes estão acompanhando de perto os desenvolvimentos de ferramentas como OpenCost e Kubecost que permitem a visibilidade de custos para equipes de produto e plataforma em organizações que adotaram o Kubernetes. Nestas fases iniciais, eles acreditam que o OpenCost ainda não funciona bem com certos workloads, como instâncias de curta duração (spot instances) frequentemente usadas em pipelines de dados.

79. OpenRewrite

Avalie

Nos deparamos com vários casos de uso para ferramentas de inteligência de código: migrar para uma nova versão de API de uma biblioteca amplamente usada, entender o impacto de uma vulnerabilidade recém-descoberta em uma determinada biblioteca de uma empresa, ou aplicar atualizações a muitos serviços que foram criados a partir do mesmo modelo. Sourcegraph ainda é uma ferramenta popular neste ecossistema, e **OpenRewrite** é uma outra ferramenta que queremos destacar. Embora nossas equipes tenham usado principalmente em Java para problemas específicos, como atualizar serviços criados por meio de um kit de inicialização, ele continua a ampliar sua cobertura de linguagens e casos de uso. Gostamos que ele vem com um catálogo de receitas, que descrevem as alterações a

serem feitas, por exemplo, para migrar frameworks comumente usados entre versões. O mecanismo de refatoração, as receitas incluídas e os plugins da ferramenta de compilação são software de código aberto, o que torna mais fácil para as equipes usarem o OpenRewrite apenas quando precisam. Nos resta saber como o amadurecimento do ecossistema de ferramentas de inteligência de código, que são todas baseadas na análise do código-fonte em uma árvore de sintaxe abstrata (AST), será impactado pelos rápidos avanços no ecossistema de LLMs.

80. OrbStack

Avalie

OrbStack é uma forma de executar contêineres Docker no macOS; nossas pessoas desenvolvedoras o consideraram mais leve, mais rápido e mais simples de configurar e usar em comparação ao Docker Desktop e o Colima. A ferramenta ainda está em desenvolvimento e, por isso, possui menos recursos, mas já demonstra um grande potencial devido à sua simplicidade e velocidade. Você também pode usar o OrbStack para criar e gerenciar VMs Linux no macOS.

81. Pixie

Avalie

Pixie é uma ferramenta de observabilidade para aplicações nativas de Kubernetes. Ela adota uma abordagem interessante para a observabilidade ao aproveitar o eBPF para coletar automaticamente dados de telemetria de várias fontes de dados. Os dados de telemetria coletados são armazenados localmente em cada nó e processados centralmente por meio de sua API de plano de controle. No geral, achamos que Pixie vale a pena ser avaliado para observabilidade no ecossistema Kubernetes.

82. Tabnine

Avalie

Tabnine é um dos atuais concorrentes no movimentado ecossistema de assistentes de programação. Ele fornece sugestões de preenchimento de código em linha e bate-papo diretamente na IDE. Similar ao GitHub Copilot, o Tabnine existe bem antes da expectativa exagerada atual e, portanto, é um dos produtos mais avançados do setor. Ao contrário do Copilot, ele usa um modelo que é treinado apenas em código licenciado de forma permissiva e oferece uma versão para auto-hospedagem pensado para atender as organizações que se preocupam em enviar seus trechos de código para serviços de terceiros. O Tabnine está disponível em duas versões: uma gratuita que é limitada; e uma paga que oferece sugestões mais abrangentes e também oferece um modo com um modelo local (embora menos poderoso), que você pode usar sem precisar de conexão com a internet.

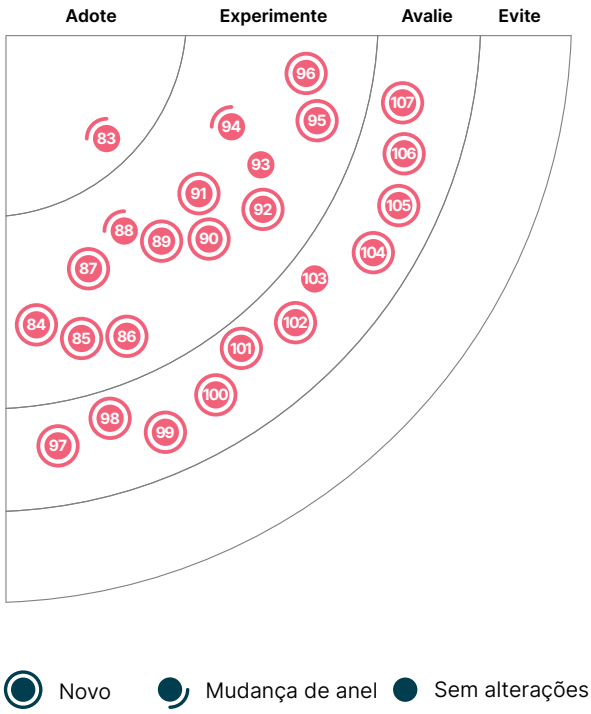
Linguagens e Frameworks

Adote
83. Playwright

Experimente
84. APIs Mínimas do .NET
85. Ajv
86. Armeria
87. AWS SAM
88. Dart
89. fast-check
90. Kotlin com Spring
91. Mockery
92. Netflix DGS
93. OpenTelemetry
94. Polars
95. Pushpin
96. Snowpark

Avalie
97. Perfis de Referência
98. GGML
99. GPTCache
100. API de Inflexão Gramatical
101. htmx
102. Kotlin Kover
103. LangChain
104. LlamaIndex
105. promptfoo
106. Semantic Kernel
107. Spring Modulith

Evite
—



83. Playwright

Adote

Com **Playwright,** você pode escrever testes de ponta a ponta que são executados no Chrome, Firefox e WebKit. Ao usar o Chrome DevTools Protocol (CDP), o Playwright pode oferecer novos recursos e eliminar muitos dos problemas encontrados com o WebDriver. Os navegadores baseados em Chromium implementam o CDP diretamente. No entanto, para suportar o Firefox e o WebKit, a equipe do Playwright precisa enviar patches para esses navegadores, o que pode às vezes limitar o framework. Os muitos recursos do Playwright incluem: esperas automatizadas (auto-waits) integradas, que resultam em testes mais confiáveis e mais fáceis de entender; Contextos do navegador (browser contexts), que permitem testar se a persistência de sessão entre abas está funcionando corretamente; e a capacidade de simular notificações, geolocalização e configurações de modo escuro. Nossas equipes estão impressionadas com a estabilidade que o Playwright traz para o conjunto de testes e gostam de obter feedback mais rapidamente ao executar testes em paralelo. Outros recursos que diferenciam o Playwright incluem um melhor suporte para carregamento lento e rastreamento. Embora o Playwright tenha algumas limitações — o suporte a componentes é experimental, por exemplo — nossas equipes o consideram sua primeira escolha em frameworks de teste e, em alguns casos, estão migrando do Cypress e do Puppeteer para ele.

84. APIs Mínimas do .NET

Experimente

ASP.NET core MVC provou ser uma abordagem poderosa e flexível para a construção de aplicações web que hospedam APIs. No entanto, sua flexibilidade traz consigo uma certa complexidade, incluindo boilerplate e convenções que nem sempre são óbvias. O roteamento fornecido pelo ASP.NET permite que múltiplos serviços sejam hospedados em uma única aplicação, mas no mundo atual de funções sem servidor e microsserviços que podem ser implantados de forma independente, essa flexibilidade pode ser um exagero. **APIs Mínimas do .NET** fornecem uma abordagem simples para implementar uma aplicação web de única API no ecossistema .NET. O framework API Mínima pode implementar um endpoint de API com apenas algumas linhas de código. O API Mínima se junta à nova geração de frameworks de API - incluindo Micronaut, Quarkus e Helidon - que são otimizados para implantações leves e tempos de inicialização rápidos. Estamos interessados na combinação de APIs Mínimas e
.NET 7 Native AOT para a implementação de microsserviços simples e leves em funções sem servidor.

85. Ajv

Experimente

Ajv é uma popular biblioteca JavaScript usada para validar um objeto de dados contra uma estrutura definida usando um schema JSON. Ajv é rápido e flexível para validar tipos de dados complexos. Ele suporta uma ampla gama de recursos de schema, incluindo palavras-chave e formatos personalizados. É usado por muitos aplicativos e bibliotecas JavaScript de código aberto. Nossas equipes usaram Ajv para implementar testes de contrato do consumidor em pipelines de CI, e, junto com outras ferramentas para gerar dados simulados com o schema JSON, é um recurso muito poderoso. No mundo do TypeScript, Zod é uma alternativa popular que tem uma API declarativa para definir o schema e validar os dados.

86. Armeria

Experimente

Armeria é um framework de código aberto para construção de microsserviços. Nossas equipes o usaram para construir APIs assíncronas, e nós gostamos bastante da abordagem de lidar com as preocupações transversais, como distributed tracing ou circuit breakers, por meio de service decorators. O framework inclui a reutilização de portas para o tráfego tanto gRPC quanto REST, entre outras escolhas de design inteligentes. Com Armeria, podemos adicionar novos recursos incrementalmente em gRPC sobre um código existente em REST ou vice-versa. No geral, achamos Armeria um framework de microsserviços flexível com várias integrações prontas para uso.

87. AWS SAM

Experimente

AWS Serverless Application Model (SAM) é um framework de código aberto para construir aplicações sem servidor na infraestrutura de nuvem AWS. Anteriormente, mencionamos o Serverless Framework como um framework popular para implantar serviços sem servidor em vários provedores de nuvem, principalmente serviços baseados em AWS Lambda. O AWS SAM ganhou popularidade nos últimos tempos, devido a evolução contínua, desde seus primeiros dias. Nossas equipes acreditam ser muito fácil de configurar e também o utilizam para testar e depurar serviços baseados em AWS Lambda, incluindo execuções locais de Lambdas para desenvolvimento.

88. Dart

Experimente

Dart é uma linguagem de programação desenvolvida pelo Google que apoia a construção de aplicativos em várias plataformas, incluindo navegadores web, WebAssembly, desktop e aplicativos móveis. Sua adoção foi impulsionada pelo domínio do Flutter — um kit popular de ferramentas multiplataforma para desenvolvimento de IU impulsionada pelo Dart — no espaço de frameworks para aplicativos nativos multiplataforma. Em resposta ao feedback da comunidade, o Dart evoluiu desde suas versões iniciais e adicionou segurança de valor nulo embutida (sound null safety) na versão três e um sistema de tipos robusto. Além disso, o ecossistema do Dart está crescendo rapidamente, com uma comunidade ativa e uma variedade de bibliotecas e ferramentas disponíveis, tornando-o atrativo para pessoas desenvolvedoras.

89. fast-check

Experimente

fast-check é uma ferramenta de teste baseada em propriedades para JavaScript e TypeScript, capaz de gerar dados de teste automaticamente, para que uma ampla gama de entradas possa ser explorada sem a necessidade de criar testes separados. Isso torna mais fácil descobrir cenários de borda. Nossas equipes estão relatando bons resultados usando fast-check em testes de back-end devido à sua boa documentação, facilidade de uso e integração perfeita com frameworks de teste existentes, o que aumenta a eficiência de testes unitários.

90. Kotlin com Spring

Experimente

Há cinco anos atrás, movemos Kotlin para o anel Adote, e hoje muitas de nossas equipes relatam que Kotlin não é apenas a escolha padrão na JVM, mas que substituiu o Java quase por completo nos softwares que elas escrevem. Ao mesmo tempo, o desejo de seguir a tendência de microsserviços (microservice envy) parece estar desaparecendo — notamos que as pessoas estão começando a explorar arquiteturas com unidades de implantação maiores, usando frameworks como o Spring Modulith entre outros. Estamos cientes dos diversos frameworks bons e nativos de Kotlin e já mencionamos alguns deles anteriormente; entretanto, em alguns casos, a maturidade e a riqueza de recursos do framework Spring são grandes vantagens, e temos usado **Kotlin com Spring** com sucesso, geralmente sem ou com problemas de pequena importância.

91. Mockery

Experimente

Mockery é uma biblioteca madura do Golang que ajuda a gerar mocks para implementações de interfaces e simula o comportamento de dependências externas. Com métodos de tipagem segura para gerar expectativas de chamadas e maneiras flexíveis de simular valores de retorno, ele permite que os testes se concentrem na lógica de negócio, em vez de se preocuparem com a corretude das dependências externas. Mockery usa geradores Go e simplifica a geração e o gerenciamento dos mocks no conjunto de testes.

92. Netflix DGS

Experimente

Adotamos o GraphQL para agregação de recursos do lado do servidor e implementamos o lado do servidor usando uma variedade de tecnologias. Para serviços escritos com Spring Boot, nossas equipes tiveram boas experiências com **Netflix DGS**. Ele é construído em cima do graphql-java e fornece recursos e abstrações no modelo de programação Spring Boot que facilitam a implementação de endpoints GraphQL e a integração com recursos do Spring, como o Spring Security. Embora seja escrito em Kotlin, o DGS também funciona bem com Java.

93. OpenTelemetry

Experimente

Temos usado **OpenTelemetry** como solução há algum tempo e já recomendamos experimentá-lo em edições anteriores. Sua capacidade de capturar, instrumentar e gerenciar dados de telemetria de forma contínua em vários serviços e aplicativos melhorou nosso stack de observabilidade. A flexibilidade e compatibilidade do OpenTelemetry com ambientes diversos o tornaram uma valiosa adição ao nosso kit de ferramentas.

Estamos agora particularmente curiosas sobre o recente lançamento da especificação do Protocolo OpenTelemetry (OTLP), que inclui gRPC e HTTP. Este protocolo padroniza o formato e a transmissão de dados de telemetria, promovendo a interoperabilidade e simplificando as integrações com outras ferramentas de monitoramento e análise. À medida que continuamos a explorar o potencial de integração do protocolo, estamos avaliando seu impacto de longo prazo em nossa estratégia de monitoramento e observabilidade e no cenário geral de monitoramento.

94. Polars

Experimente

Polars é uma biblioteca de dataframes em memória implementada em Rust. Ao contrário de outros dataframes (como pandas), Polars é multithread, suporta execução preguiçosa e é seguro para operações paralelas. Os dados em memória são organizados no formato Apache Arrow para operações analíticas eficientes e para permitir a interoperabilidade com outras ferramentas. Se você conhece pandas, já pode começar com os bindings Python do Polars. Acreditamos que Polars, com sua implementação em Rust e bindings Python, é um dataframe em memória de alto desempenho para suas necessidades analíticas. Nossas equipes continuam a ter uma boa experiência com o Polars, razão pela qual estamos movendo-o para Experimente.

95. Pushpin

Experimente

Pushpin é um proxy reverso que atua como intermediário entre clientes e servidores back-end que lidam com conexões de longa duração, como WebSockets e Server-Sent Events. Ele fornece uma maneira de terminar conexões de longa duração dos clientes e significa que o resto do sistema pode ser abstraído dessa complexidade. Ele gerencia de forma eficaz um grande número de conexões persistentes e as distribui automaticamente por vários servidores back-end, otimizando o desempenho e a confiabilidade. Nossa experiência usando isso para comunicação em tempo real com dispositivos móveis usando WebSockets tem sido boa, e conseguimos escalar horizontalmente para milhões de dispositivos.

96. Snowpark

Experimente

Snowpark é uma biblioteca para consultar e processar dados em escala no Snowflake. Nossas equipes o utilizam para escrever códigos gerenciáveis para interagir com dados que residem no Snowflake — é semelhante a escrever código Spark, mas para Snowflake. Em resumo, é um mecanismo que traduz código em SQL, o que o Snowflake entende. Você pode criar aplicações que processam dados no Snowflake sem mover dados para o sistema onde o código da sua aplicação é executado. Um ponto de atenção: o suporte a testes de unidade ainda está abaixo do ideal; nossas equipes compensam isso escrevendo outros tipos de testes.

97. Perfis de Referência

Avalie

Perfis de Referência — que não deve ser confundido com Perfil de Referência do Android — são perfis do Android Runtime (ART) que orientam a compilação antecipada (AOT). Eles são criados uma vez por versão em uma máquina de desenvolvimento e são enviados com o aplicativo, tornando-os disponíveis de forma mais rápida em comparação com a dependência Perfis da Nuvem, uma tecnologia mais antiga e relacionada. O runtime usa o Perfil de Referência em um aplicativo ou biblioteca para otimizar caminhos de código importantes, o que melhora a experiência para as pessoas usuárias novas ou antigas, quando o aplicativo é baixado ou atualizado. A criação de Perfis de Referência é relativamente simples e pode melhorar o desempenho de forma significativa (de até 30%) de acordo com sua documentação.

98. GGML

Avalie

GGML é uma biblioteca C para aprendizado de máquina que permite a inferência de CPU. Essa biblioteca define um formato binário para a distribuição de modelos de linguagem de grande porte (LLMs). Para fazer isso ela utiliza quantização, uma técnica que permite que os LLMs sejam executados em um hardware de consumo com inferência de CPU eficaz. A GGML suporta diferentes estratégias de quantização (por exemplo, quantização 4 bits, 5 bits, e 8 bits), e para cada uma oferece diferentes trade-offs entre eficiência e desempenho. Uma maneira rápida de testar, executar e desenvolver aplicativos com esses modelos quantizados é uma binding Python chamado C Transformers. Este é um wrapper Python no topo do GGML que elimina o código boilerplate para inferência, fornecendo uma API de alto nível. Exploramos essas bibliotecas para construir provas de conceito e experimentos. Se você estiver considerando LLMs auto-hospedados, avalie de forma cautelosa essas bibliotecas apoiadas pela comunidade para a sua organização.

99. GPTCache

Avalie

GPTCache é uma biblioteca de cache semântica para modelos de linguagem de grande porte (LLMs). Percebemos a necessidade de uma camada de cache na frente dos LLMs por dois motivos principais: melhorar o desempenho geral, reduzindo as chamadas de API externas; e reduzir o custo de operação, fazendo cache de respostas semelhantes. Ao contrário das abordagens de cache tradicionais que procuram por correspondências exatas, as soluções de cache baseadas em LLMs requerem correspondências semelhantes ou relacionadas para as consultas de entrada. O GPTCache aborda isso com a ajuda de algoritmos de embedding para converter as consultas de entrada em embeddings e, em seguida, usar um datastore vetorial para a busca de similaridades nessas embeddings. Uma desvantagem desse design, é a possibilidade de encontrar falsos positivos durante os hits de cache ou falsos negativos durante as misses de cache, razão pela qual recomendamos que você avalie cuidadosamente o GPTCache para suas aplicações baseadas em LLMs.

100. API de Inflexão Gramatical

Avalie

Em muitas línguas, o uso de gênero é mais visível do que em inglês, e as palavras mudam de acordo com o gênero. Por exemplo, o tratamento das pessoas usuárias pode exigir que as palavras sejam flexionadas, mas para esta prática, a forma masculina é o padrão. Há evidências de que isso tem um impacto negativo na performance e na atitude — e, claro, é indelicado. As soluções alternativas usando linguagem neutra em relação ao gênero podem muitas vezes parecer estranhas. Portanto, o tratamento correto das pessoas usuárias é a opção preferida, e com a **API de Inflexão Gramatical**, incorporada ao Android 14, pessoas desenvolvedoras de Android agora têm um caminho mais fácil para fazer isso.

101. htmx

Avalie

htmx é uma pequena e elegante biblioteca de interface do usuário (UI) em HTML que recentemente se tornou popular, aparentemente do nada. Durante as nossas discussões sobre o Radar, descobrimos que seu predecessor intercooler.js existia há dez anos. Ao contrário de outras estruturas JavaScript/TypeScript cada vez mais complexas e pré-compiladas, o htmx incentiva o uso direto de atributos HTML para acessar operações como AJAX, transições CSS, WebSockets e Eventos Enviados pelo Servidor. Não há nada tecnicamente sofisticado no htmx, mas sua popularidade lembra a simplicidade do hipertexto no começo da web. O site do projeto também apresenta alguns ensaios perspicazes (e divertidos) sobre hipermídia e desenvolvimento web, o que sugere que a equipe por trás do htmx pensou cuidadosamente sobre seu propósito e filosofia.

102. Kotlin Kover

Avalie

Kotlin Kover é um conjunto de ferramentas de cobertura de código projetado especificamente para Kotlin, com suporte à JVM de Kotlin, Multiplatform e projetos Android. A importância da cobertura de código está em sua capacidade de destacar segmentos não testados, o que reforça a confiabilidade do software. À medida que o Kover evolui, ele se destaca por sua capacidade de produzir relatórios HTML e XML abrangentes, junto a uma precisão incomparável adaptada ao Kotlin. Para equipes profundamente enraizadas em Kotlin, recomendamos avaliar o Kover para aproveitar seu potencial na melhoria da qualidade do código.

103. LangChain

Avalie

LangChain é um framework para construção de aplicações com modelos de linguagem de grande porte (LLMs). Para construir produtos práticos com LLMs, é necessário combiná-los com dados de usuário ou de domínio específicos que não foram parte do treinamento. LangChain preenche essa lacuna com recursos como gerenciamento de prompts, encadeamento, agentes e carregadores de documentos. O benefício de componentes como modelos de prompt e carregadores de documentos é que eles podem acelerar seu tempo de lançamento. Embora seja uma escolha popular para implementar aplicações de Geração Aumentada por Recuperação e o padrão de ReAct prompting, LangChain tem sido criticado por ser difícil de usar e muito complicado. Ao escolher um stack de tecnologia para sua aplicação com LLM, talvez seja interessante continuar procurando por frameworks similares — como Semantic Kernel — nesta área em rápida evolução.

104. LlamaIndex

Avalie

LlamaIndex é um framework de dados projetado para facilitar a integração de dados privados ou de domínio específico com modelos de linguagem de grande porte (LLMs). Ele oferece ferramentas para ingerir dados de fontes diversas, incluindo APIs, bancos de dados e PDFs, e estruturar esses dados em um formato que os LLMs possam consumir facilmente. Por meio de vários tipos de "motores", o LlamaIndex permite interações de linguagem natural com esses dados estruturados, tornando-os acessíveis para aplicações que fazem recuperação de informação baseada em consulta até àquelas que usam interfaces conversacionais. Similar ao LangChain, o objetivo do LlamaIndex é acelerar o desenvolvimento com LLMs, mas ele adota uma abordagem mais de framework de dados.

105. promptfoo

Avalie

promptfoo é uma ferramenta que permite testes de engenharia de prompts (prompt engineering) de forma automatizada. Ao integrar LLMs (modelos de linguagens de grande porte) em aplicações, o ajuste dos prompts para produzir saídas ótimas e consistentes pode ser demorado. Você pode usar o promptfoo como CLI (interface de linha de comando) ou uma biblioteca para testar os prompts sistematicamente contra casos de teste predefinidos. O caso de teste, juntamente com as afirmações, pode ser configurado em um arquivo de configuração YAML simples. Essa configuração inclui os comandos que estão sendo testados, o provedor do modelo, as verificações e os valores das variáveis que serão substituídas nos comandos. promptfoo suporta muitas verificações, incluindo verificação de igualdade, estrutura JSON, semelhança, funções personalizadas ou até mesmo o uso de um LLM para classificar as saídas do modelo. Se você está procurando automatizar o feedback sobre a qualidade dos prompts e dos modelos, avalie o promptfoo.

106. Semantic Kernel

Avalie

Semantic Kernel é a versão de código aberto de um dos componentes principais da suíte de produtos Copilot da Microsoft. É uma biblioteca Python que ajuda você a criar aplicativos que usam modelos de linguagem de grande porte (LLMs), semelhante ao LangChain. O conceito central do Semantic Kernel é seu planejador, que permite que você construa agentes movidos a LLM que criam um plano para uma usuária e o executam passo a passo com a ajuda de plugins.

107. Spring Modulith

Avalie

Fomos defensoras iniciais dos microserviços e observamos o padrão sendo usado com sucesso em inúmeros sistemas, mas também vimos os microserviços sendo mal aplicados e abusados, muitas vezes como resultado do desejo de seguir a tendência de microserviços envy. Em vez de iniciar um novo sistema com uma coleção de processos implantados separadamente, geralmente é recomendável começar com um monolito bem dimensionado e só separar unidades implantáveis separadamente quando a aplicação atingir uma escala em que os benefícios dos microserviços compensem a complexidade adicional inerente aos sistemas distribuídos. Recentemente, vimos um ressurgimento do interesse por essa abordagem e uma definição mais detalhada do que constitui exatamente um monolito bem dimensionado. **Spring Modulith** é um framework que ajuda a estruturar seu código de uma forma que facilite a separação em microserviços quando for a hora certa. Ele fornece uma maneira de modularizar seu código para que os conceitos lógicos de domínios e contexto delimitados estejam alinhados com os conceitos físicos de arquivos e estrutura de pacotes. Esse alinhamento torna mais fácil refatorar o monolito quando necessário e testar domínios de forma isolada. Spring Modulith fornece um mecanismo interno de processamento de eventos que ajuda a desacoplar ainda mais os módulos dentro de um único aplicativo. O melhor de tudo é que ele se integra ao ArchUnit e ao jmolecules para automatizar a verificação de suas regras de design orientado a domínio.

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

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

Assine

A Thoughtworks é uma consultoria global de tecnologia que integra estratégia, design e engenharia de software para alavancar a inovação digital. Somos mais de 11,5 mil pessoas distribuídas entre 51 escritórios e em 18 países. Há mais de 30 anos, trabalhamos junto a nossas clientes para criar impacto extraordinário, usando a tecnologia como diferenciador para ajudá-las a resolver problemas de negócio complexos.

/thoughtworks

Strategy. Design. Engineering.