ThoughtWorks®

TECHNOLOGY RADAR VOL. 20

Um guia com opiniões firmes sobre as fronteiras da tecnologia

thoughtworks.com/radar
#TWTechRadar

CONTRIBUIÇÕES

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

Esta edição do Technology Radar da ThoughtWorks teve como base um encontro do Conselho Consultivo de Tecnologia, que se reuniu em Shenzhen em Março de 2019

Rebecca Parsons
(Diretora de Tecnologia)

Martin Fowler
(Cientista-chefe)

Bharani Subramaniam

Erik Dörnenburg

Evan Bottcher

Fausto de la Torre

Hao Xu

Ian Cartwright

James Lewis

Jonny LeRoy

Ketan Padegaonkar

Lakshminarasimhan Sudarshan

Marco Valtas

Mike Mason

Neal Ford

Ni Wang

Rachel Laycock

Scott Shaw

Shangqi Liu

Zhamak Dehghani

Tradução: Alexey Villas Boas, Camilla Crispim, Daniela Araujo, Enzo Zuccolotto, Glauco Vinicius, Hellen Guareschi, Marco Valtas, Paula Ribas e Ricardo Cavalcanti

SOBRE O RADAR

'ThoughtWorkers' são pessoas apaixonadas por tecnologia. Nós desenvolvemos, pesquisamos, testamos, contribuímos com código livre, escrevemos 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 para a empresa e as tendências tecnológicas que impactam significativamente a nossa indústria.

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

Nós encorajamos você a explorar essas tecnologias para obter mais detalhes. O Radar é gráfico por natureza, agrupando os itens em técnicas, ferramentas, plataformas e linguagens & frameworks. Quando itens do radar puderem ser classificados em mais de um quadrante, escolhemos aquele que parece mais adequado. Além disso, agrupamos esses itens em quatro anéis para refletir a nossa posição atual em relação a eles.

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

RADAR EM UM RELANCE

1 ADOTE

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

2 EXPERIMENTE

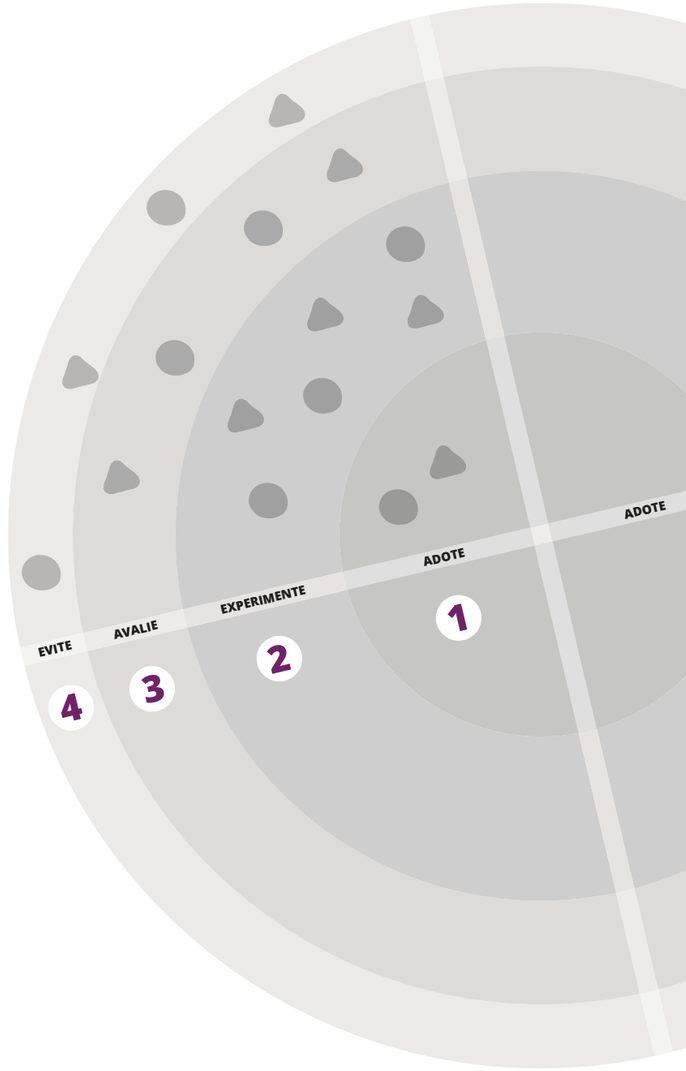
Vale a pena ir atrás. É importante entender como desenvolver essa capacidade. As empresas devem experimentar esta tecnologia em um projeto que possa lidar com o risco.

3 AVALIE

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

4 EVITE

Prossiga com cautela.



▲ NOVO OU MODIFICADO
● SEM MODIFICAÇÃO

Itens novos ou que sofreram alterações significativas desde o último Radar são representados como triângulos, enquanto os itens que não mudaram são representados como círculos.

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

O QUE HÁ DE NOVO?

Temas em destaque nessa edição:

As diversas formas de dados

Há dez anos, dados eram sinônimo de bancos de dados relacionais. Agora, dados podem ter uma impressionante variedade de formatos, incluindo NoSQL, séries temporais, SQL stores, como CockRoachDB e Spanner, que oferecem consistência global, assim como fluxos de eventos que oferecem recursos de consultas para arquivos de log agregados. Esse movimento é impulsionado pelo desejo das empresas por respostas em tempo real para fontes de dados cada vez maiores, mais variadas e rápidas. Para pessoas desenvolvedoras, entender as diferenças inerentes a cada tipo de uso de dados pode trazer desafios. Pessoas que trabalham com arquitetura e desenvolvimento de software devem buscar novas capacidades oferecidas pelas ferramentas e paradigmas, ao mesmo tempo assegurando-se de não fazer mau uso de novas ferramentas ao tratá-las como ferramentas familiares. Devemos aceitar o fato de que estamos no meio de uma grande mudança no cenário de dados e ainda procurando pelas estratégias e ferramentas adequadas.

'Terraformando' um ecossistema

Pessoas desenvolvedoras adoram camadas de abstração, por razões óbvias: ao encapsular a complexidade em uma abstração, elas podem se concentrar em assuntos de níveis mais altos. Vimos essa evolução em muitas edições do Radar, na maneira como os times lidam com as interseções de nuvens e contêineres. Primeiramente, o foco em Docker e seu ecossistema. O foco então desceu na stack para Kubernetes. Agora, a principal atividade que observamos está na infraestrutura como código em geral e especificamente o ecossistema Terraform. Embora tenhamos recomendado ferramentas além de Terraform, seu uso na comunidade de provedores tem sido impressionante. Os destaques neste Radar incluem a Terratest, para testes de código de infraestrutura, e o novo provedor do GoCD, que permite que você configure o GoCD usando Terraform.

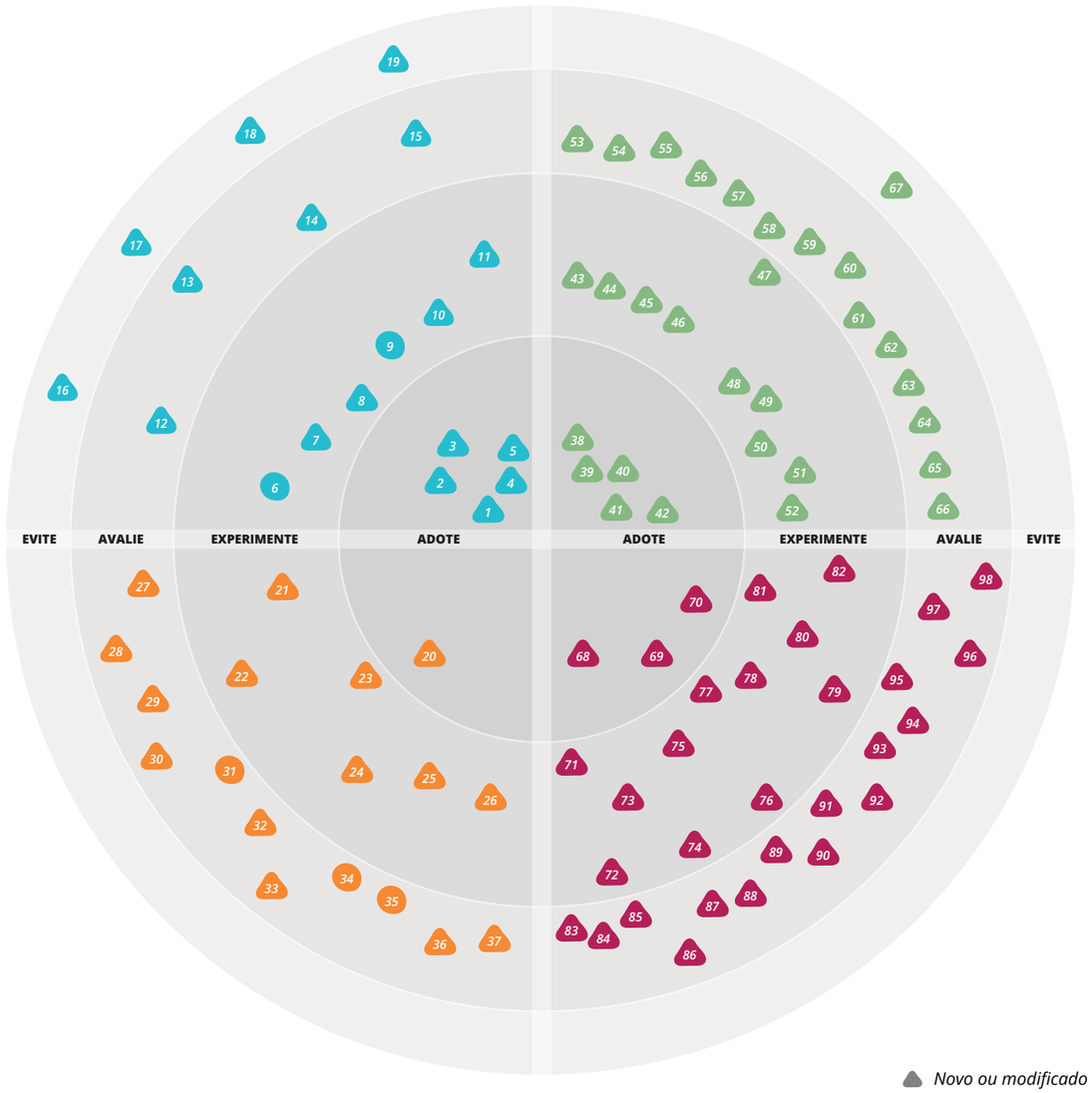
Kotlin Crescendo

Kotling, uma linguagem de código aberto criada por uma fabricante de ferramentas, continua ganhando grande espaço em nosso Radar, à medida que se expande para além de seu reduto Android. Criada internamente na JetBrains — porque não gostavam das opções disponíveis no cenário de linguagens existente —, Kotlin parece repercutir entre as pessoas desenvolvedoras em um amplo espectro. Ela continua aparecendo em plataformas e ferramentas como uma linguagem de uso geral e especialista, e cada vez mais em nosso Radar e nossos times de projeto (ex.: Ktor, MockK, Detekt, HTTP4K). É revigorante ver que design pragmático, ferramentas de ponta e um ecossistema florescendo, permitem que uma linguagem ambiciosa tenha sucesso.

Vazando limites de encapsulamento

Com o advento de "tudo como código", quase tudo — infraestrutura, segurança, compliance e operações — que era difícil de ser mudado se torna programaticamente manejável, ou seja, pessoas desenvolvedoras podem aplicar boas práticas de engenharia. Mesmo assim, muitas vezes vemos subsistemas de configuração altamente complexos ou excesso de dependência em ferramentas de orquestração visual – lógica se arrastando em arquivos de configuração, sintaxe horrível para condicionais em YAML e os muitos frameworks de orquestração que vimos em uma ampla variedade de tecnologias. Com o advento da programação poliglota, infraestrutura como código e *x*-como-serviço, os times acabam com diversos componentes que se fundem em um único sistema coeso. Assim, a lógica que deveria existir dentro de um limite de sistema vaza para as ferramentas de orquestração, arquivos de configuração e outras partes. Embora isso seja necessário algumas vezes, nós encorajamos os times a considerar cautelosamente manter tal código em lugares onde pessoas desenvolvedoras adotam testes, controle de versão, integração contínua e outras boas práticas de engenharia. Evite adicionar lógica de negócio em arquivos de configuração (e evite ferramentas que requeiram isso) e tente manter a orquestração ao mínimo necessário – e não como uma característica dominante em seu sistema.

O RADAR



TÉCNICAS

ADOTE
1. Quatro métricas fundamentais
2. Micro frontends
3. Formatação de código automatizada e opinativa
4. Programação Poliglota
5. Segredos como serviço

EXPERIMENTE
6. Engenharia de Caos
7. Varredura de segurança de contêineres
8. Entrega Contínua para modelos aprendizado de máquina (CD4ML)
9. Trituração criptografada
10. Analisador automatizado da configuração de infraestrutura
11. Malha de serviços

AVALIE
12. Ethical OS
13. Contratos inteligentes
14. Transferência de aprendizado para NLP
15. Mapeamento Wardley

EVITE
16. Utilizando notebooks Jupyter em produção
17. Perfurando o encapsulamento com captura de dados alterados
18. Trem de implantação
19. Templates em YAML

PLATAFORMAS

ADOTE
20. Contentful

EXPERIMENTE
21. AWS Fargate
22. EVM além da Ethereum
23. InfluxDB
24. Istio
25. Kafka Streams
26. Nomad

AVALIE
27. CloudEvents
28. Cloudflare Workers
29. Deno
30. Hot Chocolate
31. Knative
32. MinIO
33. Prophet
34. Quorum
35. SPIFFE
36. Tendermint
37. TimescaleDB

EVITE

FERRAMENTAS

ADOTE
38. Cypress
39. Jupyter
40. LocalStack
41. Terraform
42. Ambientes de desenvolvimento de UI

EXPERIMENTE
43. AnyStatus
44. AVA
45. batect
46. Elasticsearch LTR
47. Helm
48. InSpec
49. Lottie
50. Stolon
51. TestCafe
52. Traefik

AVALIE
53. Anka
54. Cage
55. Cilium
56. Detekt
57. Flagr
58. Gremlin
59. Honeycomb
60. Humio
61. Kubernetes Operators
62. OpenAPM
63. Systems
64. Taurus
65. Provedor Terraform para GoCD
66. Terratest

EVITE
67. CloudFormation escrito à mão

LINGUAGENS & FRAMEWORKS

ADOTE
68. Apollo
69. MockK
70. TypeScript

EXPERIMENTE
71. Apache Beam
72. Formik
73. HiveRunner
74. joi
75. Ktor
76. Laconia
77. Puppeteer
78. Reactor
79. Resilience4j
80. Room
81. Rust
82. WebFlux

AVALIE
83. Aeron
84. Arrow
85. Chaos Toolkit
86. Dask
87. Embark
88. fastai
89. http4k
90. Immer
91. Karate
92. Micronaut
93. Next.js
94. Pose
95. react-testing-library
96. ReasonML
97. Taiko
98. Vapor

EVITE

TÉCNICAS

Quatro métricas fundamentais
ADOTE

O minucioso relatório State of DevOps tem como foco uma análise estatística e orientada por dados de organizações de alta performance. O resultado dessa pesquisa de vários anos, publicado no Accelerate, mostra uma ligação direta entre a performance organizacional e a performance de entrega do software. Os pesquisadores determinaram que apenas quatro métricas fundamentais diferenciam as organizações de baixo, médio e alto desempenho: lead time, frequência de deployment, tempo médio de restauração (MTTR) e porcentagem de falha de alteração. Realmente, achamos que essas quatro métricas fundamentais são um recurso simples, porém poderoso para ajudar líderes e times a focar em medir e melhorar o que é importante. Um bom ponto de partida é equipar as pipelines de compilação para extrair essas quatro métricas fundamentais e tornar o fluxo de valor da entrega de software visível. Pipelines GoCD, por exemplo, possibilitam medir essas quatro métricas fundamentais como uma entidade de primeira classe do GoCD analytics.

Micro frontends
ADOTE

Temos visto benefícios significativos com a introdução de microsserviços, que têm permitido aos times escalar a entrega de serviços implantados e mantidos independentemente. Infelizmente, também temos visto muitos times criarem um monólito de frontend — uma aplicação para navegador grande e confusa sobre seus serviços de backend — neutralizando os benefícios dos microsserviços. Desde que apresentamos os micro frontends como uma técnica para resolver essa questão, tivemos experiências positivas com a abordagem e encontramos vários padrões para usar os micro frontends, mesmo conforme mais e mais códigos migram do servidor para o navegador web. Até agora, contudo, os componentes web têm sido confusos neste campo.

Formatação de código automatizada e opinativa
ADOTE

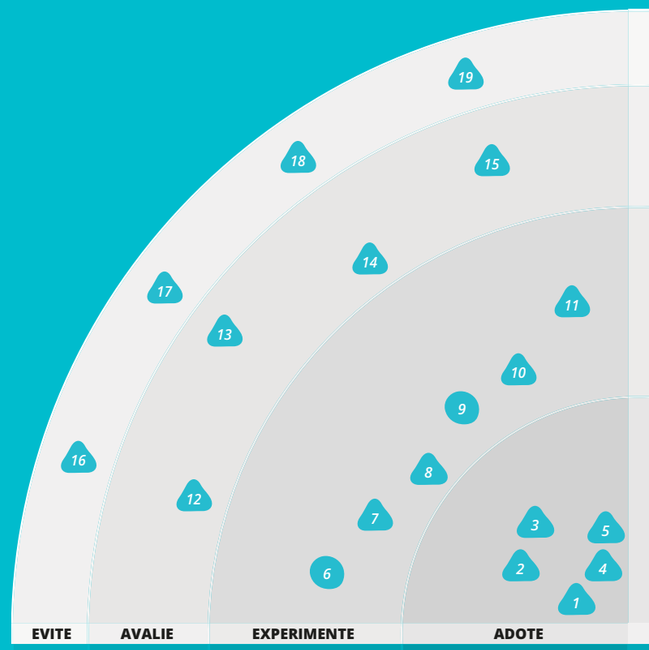
Desde que conseguimos nos lembrar, o estilo usado para formatar código tem sido uma questão de gosto pessoal, política da empresa e debates acalorados. Finalmente, a indústria parece estar se cansando dessa discussão sem fim e times estão economizando muito tempo ao abandonar essas discussões e adotar apenas ferramentas de formatação de código automatizadas e opinativas. Mesmo que você não concorde totalmente com as opiniões de várias ferramentas, focar no que o seu código faz em vez da sua aparência é algo que a maioria dos times deveria fazer. Prettier tem recebido nosso voto para JavaScript, mas ferramentas similares, como Black para Python, estão disponíveis para muitas outras linguagens e cada vez mais sendo embutidas, como

ADOTE
1. Quatro métricas fundamentais
2. Micro frontends
3. Formatação de código automatizada e opinativa
4. Programação Poliglota
5. Segredos como serviço

EXPERIMENTE
6. Engenharia de Caos
7. Varredura de segurança de contêineres
8. Entrega Contínua para modelos aprendizado de máquina (CD4ML)
9. Trituração criptografada
10. Analisador automatizado da configuração de infraestrutura
11. Malha de serviços

AVALIE
12. Ethical OS
13. Contratos inteligentes
14. Transferência de aprendizado para NLP
15. Mapeamento Wardley

EVITE
16. Utilizando notebooks Jupyter em produção
17. Perfurando o encapsulamento com captura de dados alterados
18. Trem de implantação
19. Templates em YAML



TÉCNICAS

Escolher a linguagem certa para o trabalho pode aumentar significativamente a produtividade. E dar suporte ao ecossistema de uma linguagem é importante para que sistemas entrem em operação rapidamente e pessoas desenvolvedoras tenham ferramentas adequadas para resolver problemas.

(Programação Poliglota)

O desenvolvimento tradicional de modelos de aprendizado de máquina tende a ter longos tempos de ciclo entre treinar modelos e implementá-los em produção. Por isso, muitas vezes acabamos com modelos treinados com dados (agora) obsoletos em produção. A aplicação de práticas de entrega contínua ao aprendizado de máquina pode lidar com esses dois problemas.

(Entrega Contínua para modelos aprendizado de máquina (CD4ML))

vemos com Golang e Elixir. A chave aqui é não passar horas discutindo quais regras impor, em vez disso, escolher uma ferramenta que é opinativa, minimamente configurável e automatizada – idealmente como um gancho pré-commit.

Programação Poliglota

ADOTE

Colocamos a programação poliglota em Experimente em um de nossos primeiros radares sugerindo que a escolha da linguagem correta para o trabalho poderia aumentar a produtividade significativamente, e havia novas linguagens que mereciam ser consideradas. Queremos trazer essa sugestão novamente, porque estamos vendo uma nova tendência de padronizar stacks de linguagem, tanto por pessoas desenvolvedoras quanto por empresas. Embora reconheçamos que não ter restrições nos usos da linguagem pode criar mais problemas que soluções, promover algumas linguagens que suportam diferentes ecossistemas ou funcionalidades de linguagens é importante para empresas acelerarem processos e entrarem em operação mais rapidamente, e para pessoas desenvolvedoras terem as ferramentas certas em mãos para resolverem problemas.

Segredos como serviço

ADOTE

Humanos e máquinas usam segredos por todo o fluxo de valor ao construir e operar software. As pipelines de compilação precisam de segredos para fazer interface com infraestruturas seguras como registros de contêiner, as aplicações usam chaves de API como segredos para ter acesso aos recursos de negócio, e as comunicações serviço-a-serviço são protegidas usando certificados e chaves como segredos. Você pode configurar e recuperar esses segredos de maneiras diferentes. Durante muito tempo, alertamos as pessoas desenvolvedoras sobre o uso de gerenciamento de código-fonte para armazenar segredos. Recomendamos desacoplar o gerenciamento de segredos do código-fonte e usar ferramentas como git-secrets e Talisman para evitar armazenar segredos no código-fonte. Temos usado segredos como serviço como uma técnica padrão para armazenar e acessar segredos. Com essa técnica, você pode usar ferramentas como Vault ou AWS Key Management (KMS) para ler/escrever segredos em um terminal HTTPS com níveis refinados de controle de acesso. Segredos como serviço usam provedores de identidade externos como AWS IAM para identificar atores que solicitam acesso a segredos. Esses atores se autenticam com o serviço de segredos. Para esse processo funcionar, é importante automatizar o bootstrapping de identidade dos atores, serviços e aplicações. As plataformas baseadas em SPIFFE melhoraram a automação de atribuição de identidades para serviços.

Engenharia de Caos

EXPERIMENTE

No ano passado, vimos a Engenharia de Caos passar de uma ideia muito comentada para se tornar uma abordagem aceita e estabelecida para melhorar e assegurar a resiliência de sistemas distribuídos. À medida que organizações pequenas e grandes começam a implementar a Engenharia de Caos como um processo operacional, estamos aprendendo como aplicar essas técnicas com segurança em escala. A abordagem definitivamente não é para todo mundo e, para ser efetiva e segura, requer suporte organizacional em escala. A aceitação por parte da indústria e a expertise disponível definitivamente aumentarão com a aparição de serviços comerciais, como Gremlin, e ferramentas de deployment, como Spinnaker, implementando algumas ferramentas de engenharia de caos.

Varredura de segurança de contêineres

EXPERIMENTE

A revolução do contêiner em torno do Docker reduziu massivamente o desgaste para mover aplicações entre ambientes, aumentando a adoção da entrega contínua e da implantação contínua. Essa última, especialmente, fez um estrago grande nos controles tradicionais do que pode entrar em produção. A técnica da varredura de segurança de contêineres é uma resposta necessária a isso. Ferramentas na pipeline de compilação verificam automaticamente os contêineres que fluem pela pipeline contra vulnerabilidades conhecidas. Desde que mencionamos esta técnica pela primeira vez, o cenário de ferramentas amadureceu e a técnica se provou útil nos esforços de desenvolvimento com nossos clientes.

Entrega Contínua para modelos aprendizado de máquina (CD4ML)

EXPERIMENTE

A entrega contínua para modelos de aprendizado de máquina (CD4ML) aplica as práticas de entrega contínua ao desenvolvimento de modelos de aprendizado de máquina para que estejam sempre prontos para produção. Essa técnica

soluciona dois problemas principais do modelo tradicional de aprendizado de máquina: um tempo longo de ciclo entre treinar modelos e implementá-los em produção, o que muitas vezes inclui converter manualmente o modelo para código pronto para produção, e usar modelos de produção que foram treinados com dados obsoletos. Uma pipeline de entrega contínua de um modelo de aprendizado de máquina tem dois gatilhos: (1) mudanças na estrutura do modelo; (2) mudanças no conjunto de dados de treinamento e testes. Para isso funcionar, é preciso versionar tanto os conjuntos de dados quanto o código-fonte do modelo. A pipeline frequentemente inclui passos como testar o modelo usando conjunto de dados de teste, usando a conversão automática do modelo (se necessário) com ferramentas como H2O, e implementar o modelo em produção para entregar valor.

Trituração criptografada

EXPERIMENTE

Manter um controle adequado de dados confidenciais é difícil, especialmente quando eles são copiados para fora de um sistema de registro mestre em razão de backup e recuperação. A trituração criptografada é a prática de tornar dados confidenciais ilegíveis, sobrescrevendo ou deletando deliberadamente as chaves de criptografia usadas para proteger esses dados. Considerando que há sistemas – como aplicações de auditoria ou blockchain – que não devem ou não podem deletar registros históricos, essa técnica é muito útil para proteção de privacidade e conformidade com o GDPR.

Analisador automatizado da configuração de infraestrutura

EXPERIMENTE

Já há algum tempo recomendamos um maior controle do time de desenvolvimento sobre toda sua stack, incluindo a infraestrutura. Isso significa maior responsabilidade do próprio time de desenvolvimento para configurar a infraestrutura de maneira segura, protegida e em conformidade com padrões da organização. Ao adotar estratégias de nuvem, a maioria das organizações usa como padrão uma configuração bem controlada e centralizada para reduzir riscos, mas isso também cria gargalos substanciais de produtividade. Uma abordagem alternativa é permitir que os times gerenciem sua própria configuração e usem um analisador automatizado da configuração de infraestrutura para garantir que a configuração seja definida de maneira segura e protegida. As opções incluem analisadores de código aberto, como prowler para AWS e kube-bench para Kubernetes. Para uma detecção mais contínua, vale a pena olhar para plataformas de nuvem como a AWS Config Rules, entre outros serviços comerciais.

Malha de serviços

EXPERIMENTE

A malha de serviços é uma abordagem para operar um ecossistema de microsserviços seguro, rápido e confiável. Ela tem sido um importante passo para facilitar a adoção de microsserviços em escala. Oferece descoberta, segurança, rastreamento, monitoramento e processamento de falhas. Fornece recursos interfuncionais sem a necessidade de um ativo compartilhado, como uma API gateway ou bibliotecas baking em cada serviço. Uma implementação típica envolve processos de proxy reversos leves, também conhecidos como sidecars, implementados junto com cada processo de serviço em um contêiner separado. Os sidecars interceptam o tráfego de entrada e saída de cada serviço e fornecem os recursos interfuncionais mencionados acima. Essa abordagem aliviou os times de serviços distribuídos de construir e atualizar os recursos que a malha oferece como código em seus serviços. Isso levou a uma adoção ainda mais fácil da programação poliglota em um ecossistema de microsserviços. Nossos times têm usado essa abordagem com sucesso em projetos de código aberto, como Istio, e continuaremos a monitorar outras implementações abertas de malha de serviços, como Linkerd, de perto.

Ethical OS

AVALIE

Como pessoas desenvolvedoras na ThoughtWorks, estamos cientes do papel da ética em nosso trabalho. À medida que a sociedade se torna cada vez mais dependente da tecnologia, é importante considerarmos a ética quando tomarmos decisões como times de desenvolvimento de software. Muitas ferramentas surgiram para nos ajudar a refletir sobre algumas das implicações futuras do software que estamos construindo. Entre elas estão Tarot Cards of Tech e Ethical OS, das quais tivemos bons feedbacks. Ethical OS é um framework para reflexão e um conjunto de ferramentas que guiam discussões em torno da ética na construção de software. O framework é uma

TÉCNICAS

À medida que a sociedade se torna cada vez mais dependente da tecnologia, torna-se ainda mais importante considerarmos a ética ao tomar decisões como times de desenvolvimento de software. O Ethical OS é um framework de reflexão e um conjunto de ferramentas que orienta as discussões em torno da ética na construção de software.

(Ethical OS)

TÉCNICAS

Dados imutáveis são uma coisa, mas a lógica de negócios imutável é algo totalmente diferente — pense muito antes de comprometer a lógica de negócios em um contrato inteligente.

(Contratos inteligentes)

colaboração entre o Institute for the Future e o Tech and Society Solutions Lab. É baseado em um conjunto prático de zonas de risco, como vício e economia da dopamina, além de vários cenários para guiar conversas e debates.

Contratos inteligentes

AVALIE

Quanto mais experiência adquirimos usando tecnologias de registro distribuído (DLT), mais encontramos limitações em torno do estado atual dos contratos inteligentes. Publicar contratos automatizados, irrefutáveis e irreversíveis parece ótimo na teoria. Os problemas surgem quando você leva em conta como usar técnicas modernas de entrega de software para desenvolvê-los, assim como as diferenças entre as implementações. Dados imutáveis são uma coisa, mas a lógica de negócios imutável é algo completamente diferente. É muito importante pensar se devemos incluir lógica de negócios em um contrato inteligente. Também encontramos características operacionais muito diferentes entre implementações diferentes. Por exemplo, mesmo que contratos possam evoluir, plataformas diferentes suportam essa evolução em maior ou menor grau. Nosso conselho é refletir muito antes de comprometer a lógica de negócios em um contrato inteligente e pesar os méritos de diferentes plataformas antes de fazer isso.

Transferência de aprendizado para NLP

AVALIE

A transferência de aprendizado tem sido eficaz dentro do campo da visão computacional, acelerando o tempo de treinamento de um modelo com a reutilização de módulos. Quem trabalha com aprendizado de máquina está otimista com o fato de poder aplicar as mesmas técnicas no processamento de linguagem natural (NLP) com a publicação do ULMFiT e modelos pré-treinados de código aberto e exemplos de código. Achamos que a transferência de aprendizado para NLP vai reduzir significativamente o esforço para criar sistemas que lidam com a classificação de texto.

Mapeamento Wardley

AVALIE

Geralmente temos cautela ao abordar técnicas diagramáticas, mas o mapeamento Wardley é uma abordagem interessante para iniciar conversas sobre a evolução do patrimônio de software de uma organização. Na sua forma mais simples, oferece uma visualização das cadeias de valor que existem dentro de uma organização, começando pelas necessidades de clientes e progressivamente mapeando as diferentes capacidades e sistemas usados para atender essas necessidades, bem como a evolução dessas capacidades e sistemas. O valor dessa técnica está no processo de colaboração para criação dos mapas, e não no artefato em si. Recomendamos colocar as pessoas certas para produzi-los e então tratá-los como coisas vivas e em evolução, e não artefatos completos.

Utilizando notebooks Jupyter em produção

EVITE

Os notebooks Jupyter ganharam popularidade entre cientistas de dados, que os utilizam para análises exploratórias, desenvolvimento em estágios iniciais e compartilhamento de conhecimento. Esse aumento na popularidade criou uma tendência de uso em produção de notebooks Jupyter, disponibilizando ferramentas e suporte para executá-las em escala. Embora não queiramos desencorajar ninguém a usar suas ferramentas de escolha, não recomendamos o uso de notebooks Jupyter para criar código de produção escalável, sustentável e duradouro – eles não têm controle de versão eficaz, tratamento de erros, modularidade e extensibilidade, entre outros recursos básicos necessários para a criação de código escalável e pronto para produção. Em vez disso, incentivamos pessoas desenvolvedoras e cientistas de dados a trabalharem juntas para encontrar soluções que capacitem cientistas de dados a criar modelos de aprendizado de máquina prontos para produção, usando práticas de entrega contínua com as estruturas de programação corretas. Alertamos contra o uso em produção de notebooks Jupyter para

superar ineficiências em pipelines de entrega contínua para aprendizado de máquina ou testes automatizados inadequados.

Perfurando o encapsulamento com captura de dados alterados

EVITE

A captura de dados alterados (CDC) é uma técnica poderosa para capturar as mudanças de uma base de dados de um sistema e executar ações nesses dados. Uma das maneiras mais populares de fazer isso é usar o log de transação do banco de dados para identificar mudanças, e então publicar essas mudanças diretamente em um bus de eventos que possa ser usado por outros serviços. Isso funciona muito bem para casos de uso como dividir monolitos em microsserviços, mas quando usado para integração de primeira classe entre microsserviços, leva a uma perfuração do encapsulamento e vazamento da camada de dados do serviço de origem para o contrato do evento. Já falamos sobre eventos com escopo de domínio e outras técnicas que enfatizam a importância de termos nossos eventos moldando nosso domínio adequadamente. Estamos observando alguns projetos usando CDC para publicar eventos de alteração em nível de linha e consumindo diretamente esses eventos em outros serviços. Essa perfuração do encapsulamento com captura de dados alterados pode ser um terreno escorregadio levando a integrações frágeis, e gostaríamos de alertar sobre isso com este blip.

Trem de implantação

EVITE

Temos visto organizações passarem de implantações pouco frequentes para uma cadência maior usando o conceito de trem de implantação. O trem de implantação é uma técnica para coordenar implantações em múltiplos times ou componentes que têm dependências de tempo de execução. Todas as implantações acontecem em uma agenda fixa e confiável, independente de todas as funcionalidades esperadas estarem prontas (o trem não espera por você, se você o perder, tem que esperar o próximo). Embora endossemos plenamente ter disciplina com implantações e demonstrações regulares de software funcionando, experimentamos sérios inconvenientes no médio e longo prazo, uma vez que isso reforça o acoplamento temporal em torno do sequenciamento de mudanças, e a qualidade pode diminuir à medida que os times se apressam para concluir funcionalidades. Preferimos focar nas abordagens arquiteturais e organizacionais necessárias para suportar implantações independentes. Embora o trem possa ser útil para acelerar times mais lentos, também o vemos impondo um limite máximo na velocidade com a qual times mais rápidos se movem. Acreditamos que é uma técnica que deve ser abordada com um bom grau de cautela.

Templates em YAML

EVITE

À medida que infraestruturas crescem em complexidade, cresce também a complexidade dos arquivos de configuração que as definem. Ferramentas como AWS CloudFormation, Kubernetes e Helm esperam arquivos de configuração com sintaxe JSON ou YAML, provavelmente na tentativa de deixá-los fáceis de escrever e processar. Contudo, na maioria dos casos, os times rapidamente chegam a um ponto em que têm partes que são similares, mas não iguais (por exemplo, quando o mesmo serviço precisa ser implantado em diferentes regiões com uma configuração ligeiramente diferente). Para esses casos, as ferramentas oferecem templates em YAML (ou JSON), o que causou muita frustração entre profissionais da área. O problema é que as sintaxes do JSON e do YAML requerem todos os tipos de concessões para inserir funcionalidades de template, como condicionais e loops nos arquivos. Recomendamos, em vez disso, usar uma API de uma linguagem de programação, ou, quando não for uma opção, um sistema de template em uma linguagem de programação, seja uma linguagem de uso geral, como Python, ou algo especializado, como Jsonnet.

TÉCNICAS

Usar a captura de dados alterados (CDC) para publicar eventos de alteração em nível de linha e consumir diretamente esses eventos em outros serviços pode ter como resultado integrações frágeis.

(Perfurando o encapsulamento com captura de dados alterados)

Embora endossemos plenamente ter disciplina com implantações e demonstrações regulares de software funcionando, experimentamos sérios inconvenientes no médio e longo prazo: o trem de implantação reforça o acoplamento temporal em torno do sequenciamento de mudanças, e a qualidade pode diminuir à medida que os times se apressam para concluir funcionalidades.

(Trem de implantação)

PLATAFORMAS

Contentful
ADOTE

Os sistemas de gerenciamento de conteúdo (CMS) headless estão se tornando um componente comum das plataformas digitais. Contentful é um CMS headless moderno que nossos times têm integrado com sucesso em seus fluxos de desenvolvimento. Gostamos particularmente da sua abordagem API primeiro e da implementação de CMS como código. Ele suporta primitivas poderosas de modelagem de conteúdo como código e scripts de evolução de modelo de conteúdo, que permitem tratá-lo como outros esquemas de armazenamento de dados e possibilita a aplicação de práticas de design evolucionário de bancos de dados no desenvolvimento do CMS. Sua robustez e um fluxo de novas funcionalidades, incluindo um ambiente sandbox, estão impressionando nossos times ainda mais e fizeram do Contentful nossa escolha padrão neste espaço.

AWS Fargate
EXPERIMENTE

AWS Fargate, a opção de Docker como serviço na AWS, está disponível em muitas regiões agora. É uma ótima solução para situações em que times querem executar contêineres Docker – porque as funções da AWS Lambda não são poderosas o suficiente – sem ter que gerenciar instâncias EC2 ou clusters Kubernetes. Nossos times relatam experiências geralmente positivas com Fargate. Contudo, a conveniência desses serviço gerenciado pode vir com um custo, financeiramente falando.

EVM além da Ethereum
EXPERIMENTE

A Ethereum Virtual Machine (EVM) foi projetada originalmente para a rede principal da Ethereum. Hoje em dia, contudo, a maioria dos times não quer mais reinventar o blockchain do zero. Em vez disso, querem usar EVM além da Ethereum. Vimos muitos times de blockchain optarem por fazer um fork da Ethereum (ex.: Quorum) ou implementar as especificações EVM (ex.: Burrow, Pantheon), adicionando seu próprio design. A intenção não é apenas reutilizar o design do Ethereum, mas também alavancar seu ecossistema e comunidade desenvolvedora. Para muitas pessoas desenvolvedoras, o conceito de "contrato inteligente" é quase equivalente a um contrato inteligente escrito em Solidity. Apesar de a Ethereum em si ter algumas limitações, a tecnologia em torno do ecossistema EVM está crescendo.

InfluxDB
EXPERIMENTE

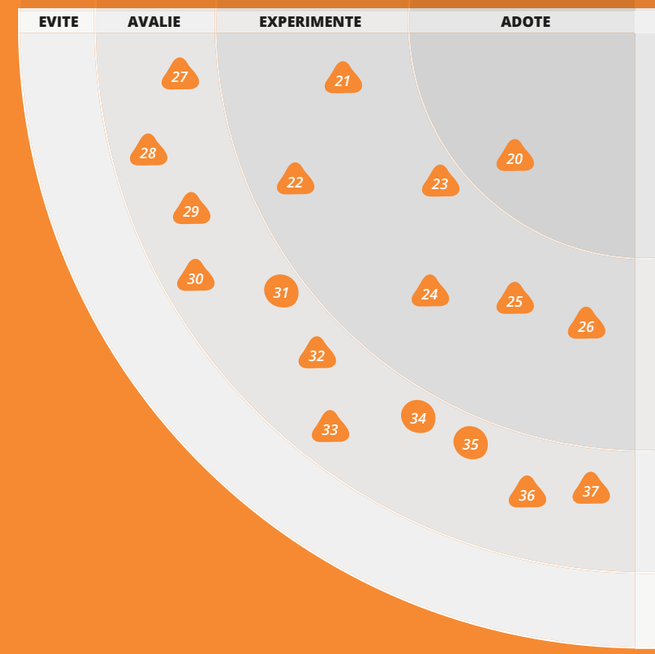
Os bancos de dados de séries temporais (TSDBs) existem há algum tempo. Entretanto, eles têm se tornado cada vez mais comuns, à medida que mais casos de uso naturalmente se adequam ao modelo de séries temporais. O InfluxDB continua a ser uma boa escolha entre TSDBs, sendo o monitoramento um

ADOTE
20. Contentful

EXPERIMENTE
21. AWS Fargate
22. EVM além da Ethereum
23. InfluxDB
24. Istio
25. Kafka Streams
26. Nomad

AVALIE
27. CloudEvents
28. Cloudflare Workers
29. Deno
30. Hot Chocolate
31. Knative
32. MinIO
33. Prophet
34. Quorum
35. SPIFFE
36. Tendermint
37. TimescaleDB

EVITE



PLATAFORMAS

Os bancos de dados de séries temporais já existem há algum tempo, mas eles têm se tornado cada vez mais comuns à medida que mais casos de uso se adequam naturalmente ao modelo de séries temporais. O InfluxDB continua a ser uma boa escolha neste espaço.

(InfluxDB)

Embora eventos sejam um mecanismo-gatilho de FaaS comum, as especificações proprietárias atuais evitam a interoperabilidade entre as nuvens. CloudEvents é um padrão em crescimento que tenta resolver esse problema.

(CloudEvents)

de seus principais casos de uso. TICK Stack é um exemplo de uma solução de monitoramento que tem o InfluxDB em seu cerne. O Influx 2.0 alfa recentemente apresentou Flux – uma linguagem de script para consulta e processamento de dados de séries temporais. Flux ainda está no começo e ainda não se pode dizer muito sobre sua adoção além do InfluxDB, mas ela promete ser mais poderosa e expressiva que a InfluxQL, e possibilita mover cargas de trabalho analíticas de séries temporais para o banco de dados. Contudo, o suporte a clusterização para o InfluxDB está disponível apenas na versão comercial, o que limitou sua adoção em alguns de nossos projetos.

Istio

EXPERIMENTE

Istio está se tornando a infraestrutura de fato para operacionalizar um ecossistema de microsserviços. Sua implementação de funcionalidades transversais – como descoberta de serviço, segurança de serviço-a-serviço e de origem-a-serviço, observabilidade (incluindo telemetria e rastreamento distribuído), rolling releases e resiliência – tem nos ajudado a iniciar implementações de microsserviços muito rapidamente. É a principal implementação da técnica de malha de serviços que temos usado. Estamos aproveitando as implantações mensais e suas melhorias contínuas com atualizações impecáveis. Usamos Istio para fazer bootstrap em nossos projetos, começando com a observabilidade (monitoramento e telemetria) e segurança de serviço-a-serviço. Estamos observando de perto suas melhorias para autenticação serviço-a-serviço tanto dentro como fora da malha. Também gostaríamos de ver Istio estabelecer boas práticas para arquivos de configuração, para encontrar um equilíbrio entre dar autonomia a pessoas desenvolvedoras de serviço e controle aos operadores da malha de serviços.

Kafka Streams

EXPERIMENTE

Kafka Streams é uma biblioteca leve para a construção de aplicações de streaming. Ela suporta APIs básicas de streaming, como juntar, filtrar, mapear e agregar, assim como armazenamento local para casos de uso comum, como janelamento e sessões. Diferente de outras plataformas de processamento de stream, como Apache Spark e Alpakka Kafka, a Kafka Streams tem sido uma boa alternativa para cenários que não necessitam de distribuição em larga escala e processamento paralelo, por isso, pudemos nos livrar de mais uma peça de infraestrutura, como agendadores de cluster. Naturalmente, Kafka Streams tem sido uma boa escolha quando operamos no ecossistema Kafka. Kafka Streams é particularmente útil quando temos que processar dados rigorosamente em ordem e exatamente uma vez. Um uso em particular de Kafka Streams é a construção de uma plataforma de alteração de captura de dados (CDC).

Nomad

EXPERIMENTE

A HashiCorp continua a lançar software interessante. Apresentamos o HashiCorp Vault em março de 2017 e as ferramentas relacionadas ao Terraform estão bastante presentes nessa edição do Radar. Movemos o Nomad para Experimente porque tivemos experiências positivas com seu uso. Embora o Kubernetes continue a ganhar popularidade, gostamos da aplicabilidade geral do Nomad. A plataforma não se limita apenas a rodar cargas de trabalho conteinerizadas, podendo também ser usada para agendar praticamente tudo. Java e Golang são suportadas nativamente, bem como agendamento de tarefas em lote e distribuído. Gostamos do foco em operações multinuvens e nuvens híbridas, algo que provavelmente se tornará mais importante para evitar nuvens grudentas, e o fato de fazer um bom agendamento.

CloudEvents

AVALIE

Fora do código de função em si, as aplicações escritas como funções sem servidor são intimamente acopladas à plataforma de nuvem em que estão hospedadas. Embora eventos sejam um mecanismo-gatilho de FaaS comum, as especificações proprietárias atuais evitam a interoperabilidade entre as nuvens. A especificação CloudEvents é um padrão crescente que foi aceito no CNCF Sandbox. O padrão ainda está em franco desenvolvimento, mas existem várias ligações de linguagem e a Microsoft anunciou um suporte de primeira classe no Azure. Esperamos que outros provedores de nuvem sigam o exemplo.

Cloudflare Workers

AVALIE

As plataformas de execução de código sem servidor ou em servidor mais modernas estão centradas em contêineres ou VMs. Contudo, Cloudflare Workers tem uma abordagem diferente para hospedar uma oferta de computação sem servidor. Ela usa V8 Isolates, o mecanismo JavaScript de código aberto desenvolvido para Chrome, para executar funções como serviço (FaaS)

em sua extensa rede CDN. O código pode ser escrito em JavaScript ou qualquer coisa que compile para WebAssembly, e dados podem ser acessados no cache da Cloudflare ou no armazenamento de chave-valor. O principal benefício para as pessoas desenvolvedoras é a performance: estando na ponta da rede, perto dos usuários finais, cold-starts levam apenas cinco milissegundos. Para o provedor, os benefícios incluem tanto a capacidade de empacotar densamente os isolates por causa de sua baixa sobrecarga de memória, quanto um desempenho mais rápido por meio da troca reduzida de contexto do processo. Esta é definitivamente uma abordagem intrigante para se monitorar e avaliar.

Deno

AVALIE

Como grupo, temos sentimentos conflitantes em relação à programação em JavaScript no servidor, especialmente quando o raciocínio para se fazer isso é simplesmente evitar a programação poliglota. Dito isso, se você decidir usar JavaScript ou TypeScript no servidor, dê uma olhada em Deno. Escrito por Ryan Dahl, o criador do Node.js, Deno tem o objetivo de evitar o que Ryan considera erros cometidos no Node.js. A plataforma traz um rigoroso sistema sandbox e gerenciamento embutido de dependência e pacotes, e suporta TypeScript sem necessidade de instalações ou configurações. Deno é construída usando Rust e V8.

Hot Chocolate

AVALIE

O ecossistema e a comunidade GraphQL continuam crescendo. Hot Chocolate é um servidor GraphQL para .NET (core e clássico). Ele permite construir e hospedar esquemas e, em seguida, realizar consultas neles. O time por trás do Hot Chocolate recentemente adicionou uma emenda de esquema, que permite que um ponto de entrada único faça consultas em múltiplos esquemas agregados de diferentes localizações. Embora haja muitas maneiras de usar erroneamente essa abordagem, é válido avaliar a possibilidade de adicioná-la ou não ao seu kit de ferramentas.

Knative

AVALIE

A arquitetura sem servidor popularizou o estilo de programação FaaS entre pessoas desenvolvedoras. Ela ajuda a manter o foco em problemas centrais do negócio com funções compiladas e implantadas de forma independente, que reagem a um evento, executam um processo de negócio, produzem outros eventos no processo e reduzem para zero. Historicamente, plataformas sem servidor proprietárias como AWS Lambda ou Microsoft Azure Functions tornaram possível esse paradigma de programação. Knative é uma plataforma de código aberto baseada em Kubernetes para executar cargas de trabalho de FaaS. Há algumas coisas que se destacam na Knative: seu código aberto e provedor agnóstico; o fato de implementar o fluxo de trabalho sem servidor como descrito no relatório do CNCF Serverless Working Group; sua capacidade de assegurar interoperabilidade entre serviços ao implementar uma interface de eventos consistente com a especificação do CNCF CloudEvents; e, o mais importante, ela resolve um desafio comum, de operar uma arquitetura baseada em contêiner de longa duração e um FaaS harmonizado, porém híbrido. Ela se integra facilmente tanto com Istio quanto Kubernetes. As pessoas desenvolvedoras podem, por exemplo, tirar proveito das estratégias de roll-out que o Istio implementa ao dividir o tráfego entre diferentes revisões das funções. Também podem se beneficiar da observabilidade fornecida pelo Istio não apenas em serviços de contêiner de longa duração, mas também em programas FaaS no mesmo ambiente Kubernetes. Antecipamos que a interface de eventos de código aberto da Knative vai continuar tornando possível novos recursos subjacentes e integrações de destinação de eventos.

MinIO

AVALIE

Armazenamento de objetos é uma escolha popular para armazenar dados não-estruturados e, em alguns casos, dados estruturados em nuvem. Nós desencorajamos o uso da nuvem genérica, mas se você quiser minimizar o risco de uma nuvem grudenta para armazenamento de objetos, achamos MinIO bem útil. Com uma camada de API compatível com S3, MinIO abstrai o armazenamento de objetos entre os provedores de nuvem, incluindo AWS, Azure e Google Cloud Platform (GCP), e temos usado com sucesso em produtos com infraestruturas flexíveis, de data centers a provedores de nuvem.

Prophet

AVALIE

Mesmo na era da aprendizagem profunda, modelos estatísticos ainda são importantes no suporte de decisões de negócios. Modelos de séries temporais são amplamente usados para projetar inventários, demanda, tráfego de clientes, e assim por diante. Fazer esses modelos manualmente para que sejam robustos e flexíveis tem sido normalmente o papel de

PLATAFORMAS

Se você decidir usar JavaScript ou TypeScript no servidor, dê uma olhada no Deno, que traz um sistema rigoroso de sandbox, erenciamento embutido de dependência e pacotes, e suporta TypeScript sem necessidade de instalações ou configurações.

(Deno)

A arquitetura sem servidor popularizou o estilo de programação de função-como-serviço entre pessoas desenvolvedoras. O Knative é uma plataforma de provedor agnóstico, baseada em Kubernetes, para executar cargas de trabalho de função-como-serviço.

(Knative)

PLATAFORMAS

Tendermint é um mecanismo de replicação de máquina de estado que permite a você implementar seus próprios sistemas blockchain.

(Tendermint)

especialistas em estatísticas ou grandes fornecedores de software comercial. Prophet é uma alternativa de código aberto para pacotes de projeções comerciais que pode ser programada em R ou Python. O Facebook afirma que usa Prophet internamente para projeções de negócios em escala e o disponibilizou como um pacote de código aberto para quem quiser usar. Gostamos do fato de que o Prophet elimina um pouco do tédio da construção, manutenção e manipulação de dados dos modelos, para que analistas humanos e experts no assunto possam se concentrar no que fazem de melhor.

Quorum
AVALIE

Quorum é "uma versão do Ethereum com foco em empresas" que visa fornecer permissão de rede e privacidade de transação, assim como um melhor desempenho. Um de nossos times trabalhou profundamente com o Quorum. Contudo, sua experiência até o momento não tem sido boa. Alguns desafios resultam da complexa programação de contratos inteligentes e alguns vêm do próprio Quorum. Por exemplo, ele não funciona bem com balanceadores de carga e possui um suporte apenas parcial a bancos de dados, o que leva a uma significativa sobrecarga de implantação. Encontramos alguns problemas de estabilidade e compatibilidade, especialmente em transações privadas. O Quorum recentemente chamou a atenção por causa da JPM Coin. Contudo, de uma perspectiva tecnológica, recomendamos cautela na implementação do Quorum, ao mesmo tempo em que ficamos de olho no seu desenvolvimento.

SPIFFE
AVALIE

A uniformização de identidade de serviço SPIFFE foi um importante passo para permitir soluções turnkey para criptografia ponta a ponta e autenticação mútua entre serviços. Os padrões SPIFFE são apoiados pelo OSS SPIFFE Runtime Environment (SPIRE), que fornece automaticamente identidades para serviços criptograficamente verificáveis. O Istio também usa SPIFFE por padrão. O SPIFFE permite muitos usos, incluindo conversão de identidade, autenticação de cliente OAuth, "criptografia em todos os lugares" do mTLS e observabilidade da carga de trabalho. A ThoughtWorks está trabalhando ativamente com as comunidades de Istio e SPIFFE para preencher as lacunas entre provedores de identidade de serviço legados e identidades baseadas em SPIFFE para que o mTLS possa ser usado em qualquer lugar entre os serviços, dentro de uma malha de serviços e fora dela.

Tendermint
AVALIE

A tolerância a falhas bizantina (BFT) é um dos problemas fundamentais em sistemas de criptomoedas e blockchain. Ela requer concordância global do sistema em um único valor de dados na presença de vários processos arbitrários faltosos, que incluem fraude maliciosa. Tendermint é um mecanismo de replicação de máquina de estado que permite a você implementar seus próprios sistemas blockchain. O mecanismo de consenso, Terdermint Core, assume a comunicação ponto a ponto e a parte do consenso, você precisa apenas implementar o resto da aplicação (ex.: construir a transação e verificar a assinatura criptográfica) e se comunicar com o Tendermint Core por meio da ABCI. Algumas implementações de blockchain já escolheram Tendermint como seu mecanismo de consenso.

TimescaleDB
AVALIE

Em edições anteriores do Radar, discutimos PostgreSQL para NoSQL. A maturidade e a extensibilidade do PostgreSQL levaram à criação de diversos mecanismos de persistência baseados no Postgres. Um que chamou nossa atenção é o TimescaleDB, um banco de dados que permite gravações rápidas e consultas otimizadas sobre dados de séries temporais. Embora não seja (ainda) tão completo quanto InfluxDB, o TimescaleDB oferece um modelo de dados alternativo e capacidade de consulta. Você deve avaliar o TimescaleDB se tiver necessidades de escalabilidade modestas, preferir usar SQL e prezar pela estabilidade e pela interface administrativa familiar que o PostgreSQL oferece.

FERRAMENTAS

Cypress
ADOTE

Continuamos recebendo feedback positivo sobre as ferramentas de testes de UI para web "pós-Selenium", como Cypress, TestCafe e Puppeteer. Executar testes de ponta a ponta pode apresentar desafios, como a longa duração do processo de execução, a instabilidade de alguns testes e a correção de falhas ao executar testes no modo headless. Nossos times tiveram experiências muito boas com Cypress resolvendo problemas comuns, como falta de desempenho e longo tempo de espera para carregar respostas e recursos. Cypress se tornou a ferramenta escolhida para testes E2E em nossos times.

Jupyter
ADOTE

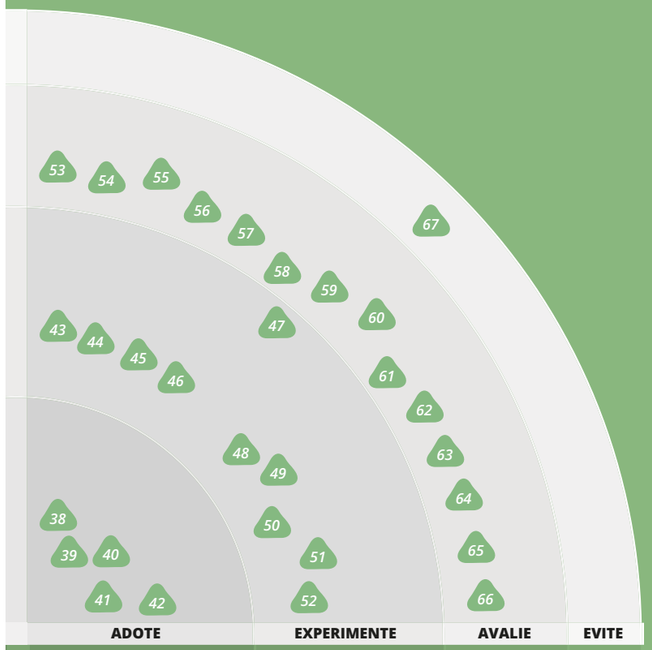
Nos últimos anos, notamos um aumento constante na popularidade de notebooks analíticos. São aplicações inspiradas em Mathematica que combinam texto, visualização e código em um documento computacional vivo. Os notebooks Jupyter são amplamente usados por nossos times para prototipagem e exploração em análise de dados e aprendizado de máquina. Colocamos o Jupyter em Adote nessa edição do Radar para mostrar que ele emergiu como padrão atual para notebooks Python. Contudo, recomendamos cautela para usar notebooks Jupyter em produção.

LocalStack
ADOTE

Um dos desafios de usar serviços de nuvem é poder desenvolver e testar localmente. LocalStack resolve esse problema para AWS, fornecendo implementações de test double locais para uma ampla variedade de serviços da AWS, incluindo S2, Kinesis, DynamoDB e Lambda. É baseada em ferramentas de ponta, como Kinesalite, dynalite e Moto, e adiciona processos isolados e funcionalidade de injeção de erros. LocalStack é muito fácil de usar e vem com um runner JUnit simples e uma extensão JUnit 5, podendo rodar dentro de um contêiner Docker. Para muitos times, ela se tornou padrão para testar serviços que são implantados na AWS.

Terraform
ADOTE

Terraform está se tornando rapidamente a escolha para criar e gerenciar infraestruturas de nuvem por meio de definições declarativas. A configuração dos servidores instanciados por Terraform é normalmente feita através de ferramentas como Puppet, Chef ou Ansible. Gostamos de Terraform porque a sintaxe de seus arquivos é bastante legível e porque suporta vários provedores de nuvem, ao mesmo tempo em que não tenta criar nenhuma abstração artificial.
A comunidade ativa adiciona suporte para



ADOTE
38. Cypress
39. Jupyter
40. LocalStack
41. Terraform
42. Ambientes de desenvolvimento de UI

EXPERIMENTE
43. AnyStatus
44. AVA
45. batect
46. Elasticsearch LTR
47. Helm
48. InSpec
49. Lottie
50. Stolon
51. TestCafe
52. Traefik

AVALIE
53. Anka
54. Cage
55. Cilium
56. Detekt
57. Flagr
58. Gremlin
59. Honeycomb
60. Humio
61. Kubernetes Operators
62. OpenAPM
63. Systems
64. Taurus
65. Provedor Terraform para GoCD
66. Terratest

EVITE
67. CloudFormation escrito à mão

FERRAMENTAS

Storybook, React Styleguidist, Compositor e MDX fornecem um ambiente abrangente para iteração rápida de componentes de UI, focando na colaboração entre designers de experiência do usuário e pessoas desenvolvedoras.

(Ambientes de desenvolvimento de UI)

Muita energia e esforços continuam a ser desperdiçados na configuração de ambientes locais de desenvolvimento e resolução de problemas do tipo "funciona na minha máquina". O batect facilita a configuração e o compartilhamento de um ambiente de compilação em Docker, lançando contêineres para executar tarefas de compilação.

(batect)

as últimas funcionalidades da maioria dos provedores de nuvem. Depois de nossa primeira (e mais cautelosa) menção a Terraform, há quase dois anos, a ferramenta continuou se desenvolvendo e evoluiu para um produto estável com um bom ecossistema, que provou seu valor em nossos projetos. O problema com o gerenciamento de estados pode agora ser contornado usando o que o Terraform chama de "backend de estado remoto". Temos usado com sucesso o AWS S3 para isso.

Ambientes de desenvolvimento de UI

ADOTE

À medida que mais times adotam DesignOps, as práticas e ferramentas nessa área também amadurecem. Os ambientes de desenvolvimento de UI fornecem ambientes abrangentes para iteração rápida de componentes de UI, focando na colaboração entre designers de experiência do usuário e pessoas desenvolvedoras. Agora temos algumas opções neste espaço: Storybook , React Styleguidist, Compositor e MDX. Você pode usar essas ferramentas de maneira independente no desenvolvimento de bibliotecas de componentes ou de sistemas de design, bem como incorporadas em um projeto de aplicação web. Muitos times conseguiram diminuir seus ciclos de feedback e melhorar o tempo do trabalho de UI na preparação para o trabalho de desenvolvimento, o que tornou o uso de ambientes de desenvolvimento de UI um padrão razoável para nós.

AnyStatus

EXPERIMENTE

Como pessoas desenvolvedoras acostumadas a fazer vários pequenos commits diariamente, confiamos nos monitores para nos notificar quando as compilações ficam verdes. AnyStatus é um aplicativo Windows leve para desktop que reúne métricas e eventos de diferentes fontes em um só lugar. Os exemplos incluem resultados e releases de compilação, health checks para diferentes serviços e métricas do sistema operacional. Pense nele como um CCTray com anabolizantes. Também está disponível como plugin do Visual Studio.

AVA

EXPERIMENTE

AVA é um executor de testes para Node.js. Embora o JavaScript seja single-threaded, IO em Node.js pode ocorrer em paralelo por sua natureza assíncrona. AVA tira vantagem disso e roda os testes concorrentemente, o que é especialmente benéfico para testes com IO pesado. Além disso, os arquivos de teste são rodados em paralelo como processos separados, proporcionando um desempenho muito melhor e um ambiente isolado para cada arquivo de teste. AVA é uma opção leve quando comparada a um framework cheio de recursos, como Jest. É opinativo e obriga você a escrever casos de testes atômicos.

batect

EXPERIMENTE

Muita energia e esforços continuam a ser desperdiçados na configuração de ambientes locais de desenvolvimento e resolução de problemas do tipo "funciona na minha máquina". Por muitos anos, nossos times adotaram o método "check out e vai", em que usamos uma abordagem de script para nos assegurar que o ambiente de desenvolvimento local está configurado de forma consistente. batect é uma ferramenta de código aberto desenvolvida por um ThoughtWorker que facilita a configuração e o compartilhamento de um ambiente de compilação baseado em Docker. batect torna-se o script de entrada para seu sistema de compilação, lançando contêineres para executar tarefas de compilação que não dependem de uma configuração local. As mudanças na configuração da compilação e dependências são compartilhadas por meio de controle de recursos sem precisar de qualquer mudança ou instalação em máquinas locais ou agentes de CI. Embora gostemos da Cage e outras ferramentas, neste espaços vemos batect crescendo rapidamente em benefícios para nossos times.

Elasticsearch LTR

EXPERIMENTE

Um dos desafios da busca é assegurar que os resultados mais relevantes para o usuário apareçam no topo da lista. É aí que Learning To Rank (LTR) pode ajudar. LTR é o processo de usar aprendizado de máquina para classificar documentos mostrados por um motor de busca. Se você usa Elasticsearch, você pode obter uma classificação de relevância de busca com o plugin Elasticsearch LTR. O plugin usa RankLib para gerar os modelos durante a fase de treinamento. Depois, ao consultar o Elasticsearch, você pode usar este plugin para "reclassificar" os resultados principais. Usamos esta ferramenta em alguns projetos e ficamos felizes com os resultados. Há também uma solução LTR equivalente para usuários de Solr.

Helm

EXPERIMENTE

Helm é um gerenciador de pacotes para Kubernetes. Ele vem com um repositório de aplicações curadas pelo Kubernetes que são mantidas no repositório de gráficos

oficial. O Helm tem dois componentes: um utilitário de linha de comando chamado Helm e um componente de cluster chamado Tiller. Proteger um cluster do Kubernetes é um tópico abrangente e cheio de nuances, mas recomendamos configurar o Tiller em um ambiente de controle de acesso baseado em função (RBAC). Usamos Helm em vários projetos de clientes e seu gerenciamento de dependências, templates e mecanismo de gancho simplificou muito o gerenciamento do ciclo de vida de aplicações no Kubernetes. Contudo, recomendamos ter atenção — o template YAML do Helm pode ser de difícil compreensão, e o Tiller ainda tem algumas arestas a serem aparadas. A expectativa é que o Helm 3 resolva esses problemas.

InSpec
EXPERIMENTE

Como uma organização lida com a autonomia de times de entrega ao mesmo tempo em que se certifica de que as soluções implantadas são seguras e estão em conformidade? Como garantir que os servidores, uma vez implantados, mantenham uma configuração consistente sem desvios? InSpec se posiciona como uma solução para conformidade e segurança contínuas, mas você pode também usá-la para testes gerais de infraestrutura. A ferramenta permite a criação de testes de infraestrutura declarativos, que podem então ser executados continuamente em ambientes provisionados, incluindo o de produção. Nossos times elogiam particularmente seu design extensível com recursos e matchers para múltiplas plataformas. Recomendamos experimentar InSpec como uma solução para o problema de assegurar a conformidade e a segurança.

Lottie
EXPERIMENTE

Uma boa animação de UI pode melhorar muito a experiência de usuário. Contudo, reproduzir uma delicada animação de designer em um aplicativo é normalmente uma tarefa desafiadora para pessoas desenvolvedoras. Lottie é uma biblioteca para Android, iOS, web e Windows que analisa animações do Adobe After Effects exportadas como JSON com Bodymovin e as renderiza nativamente para dispositivos móveis e web. Tanto designers quanto pessoas desenvolvedoras podem continuar usando suas ferramentas de costume e ainda assim colaborar com fluidez.

Stolon
EXPERIMENTE

Configurar instâncias PostgreSQL altamente disponíveis pode ser complicado, e é por isso que gostamos do Patroni – que nos ajuda a acelerar a configuração de clusters do PostgreSQL. Stolon é outra ferramenta que temos usado com sucesso para coordenar clusters de instâncias de PostgreSQL de alta disponibilidade em produção usando Kubernetes. Embora o PostgreSQL já suporte por padrão a replicação de streaming, o desafio em uma estrutura de alta disponibilidade é assegurar que clientes sempre se conectem ao master atual. Gostamos do fato de o Stolon reforçar a conexão ao master PostgreSQL correto, ao encerrar ativamente conexões a outros masters e rotear requisições para o ativo.

TestCafe
EXPERIMENTE

Temos tido boas experiências usando as ferramentas de teste de UI para web "pós-Selenium", como Cypress , TestCafe e Puppeteer. TestCafe permite que você escreva testes em JavaScript ou TypeScript e os execute no navegador. TestCafe tem vários recursos úteis, incluindo execução paralela pronta para uso e simulação de solicitação de HTTP. A ferramenta usa um modelo de execução assíncrona sem tempos de espera explícitos, o que resulta em suítes de testes muito mais estáveis. Sua API seletora facilita a implementação de padrões PageObjetc. A TestCafe recentemente lançou sua versão 1.0.x, que melhorou sua estabilidade e funcionalidade.

Traefik
EXPERIMENTE

Traefik é um proxy reverso e balanceador de carga de código aberto. Se você está procurando por um proxy de borda que forneça roteamento simples sem todos os recursos de NGINX e HAProxy, Traefik é uma boa escolha. O roteador fornece reconfiguração sem necessidade de recarga, além de métricas, monitoramento e disjuntores, essenciais para a execução de microsserviços. Ele também se integra muito bem com Let's Encrypt para fornecer SSL termination, assim como componentes de infraestrutura como Kubernetes, Docker Swarm ou Amazon ECS para selecionar automaticamente novos serviços ou instâncias e incluí-los em seu balanceamento de carga.

FERRAMENTAS

Confie, mas verifique. O InSpec ajuda a garantir que os servidores, uma vez implantados, permaneçam seguros e em conformidade durante sua vida útil operacional.

(InSpec)

FERRAMENTAS

Aproveitando o eBPF do Linux, Cilium fornece rede e segurança reconhecidas por API com base na identidade de serviço, pod ou contêiner, é dinâmico e pronto para microsserviços.

(Cilium)

Detekt é uma ferramenta de análise estática de código para Kotlin, que encontra cheiros e complexidade no código. Você pode executá-la a partir da linha de comando ou usar seus plug-ins para integração com ferramentas populares de desenvolvimento.

(Detekt)

Anka
AVALIE

Anka é um conjunto de ferramentas para criar, gerenciar e distribuir ambientes virtuais reproduzíveis de compilação e testes de macOS, para desenvolvimento iOS e macOS. Ele traz a experiência do Docker para ambientes macOS: início imediato, CLI para gerenciar máquinas virtuais e registro para versão e tag em máquinas virtuais para distribuição. Descobrimos Anka quando propusemos uma solução de nuvem privada em macOS para um cliente. Essa é uma ferramenta que vale a pena considerar quando aplicamos o fluxo de trabalho de DevOps em ambientes iOS e macOS.

Cage
AVALIE

Cage é um wrapper de código aberto em torno do Docker Compose que permite que você configure e execute múltiplos componentes dependentes como uma única aplicação. Permite também que você orquestre a execução de componentes, como imagens em Docker, código de serviço do repositório, scripts para carregar datastores e pods, que são contêineres executados juntos como um só. O Cage usa o formato de arquivo de configuração Docker Compose v2. Ele resolve algumas das lacunas do Docker Compose, como suporte a múltiplos ambientes, incluindo o ambiente de desenvolvimento para executar aplicações distribuídas na máquina local, e o ambiente de testes para executar testes de integração e produção.

Cilium
AVALIE

Abordagens tradicionais de segurança de rede Linux, como iptables, filtram endereços de IP e portas TCP/UDP. Contudo, esses endereços de IP se transformam frequentemente em ambientes dinâmicos de microsserviços. Aproveitando o eBPG do Linux, Cilium fornece rede e segurança reconhecidas por API ao inserir a segurança de maneira transparente, baseada em serviço, pod ou contêiner, em contraste com a identificação de endereço de IP. Ao desacoplar a segurança do endereço, o Cilium pode ter um papel significativo como uma nova camada de proteção de rede, e recomendamos que você avalie.

Detekt
AVALIE

Detekt é uma ferramenta de análise estática de código para Kotlin, que encontra cheiros e complexidade no código. Você pode executá-la a partir da linha de comando ou usar seus plug-ins para integração com ferramentas populares de desenvolvimento, como Gradle (para fazer análise do código em compilações) ou SonarQube (para fazer cobertura de código, além da análise estática de código) e IntelliJ. Detekt é uma ótima adição para pipelines de compilação de aplicações Kotlin.

Flagr
AVALIE

Feature toggles são uma importante técnica em cenários de implantação contínua. Nós nos deparamos com várias boas soluções feitas em casa, mas gostamos da abordagem da Flagr: uma feature toggle completa como um serviço e distribuída como um contêiner Docker. Ela vem com SDKs para todas as principais linguagens, tem uma API REST simples e bem documentada e fornece um frontend conveniente.

Gremlin
AVALIE

Gremlin é uma solução de software como serviço para organizações conduzirem experimentos de caos e testarem a resiliência de seus sistemas. Ela vem com uma série de ataques de falha – incluindo falhas de recursos, rede e estado – que podem ser executados ad hoc ou com agendamento e requer uma preparação mínima (especialmente para usuários de Kubernetes, que podem executar Helm para instalar o Gremlin). O cliente Gremlin também tem uma boa interface do usuário baseada na web, o que torna fácil executar e gerenciar experimentos de caos.

Honeycomb
AVALIE

Honeycomb é uma ferramenta de observabilidade que consome dados ricos de sistemas de produção e os torna gerenciáveis por meio de amostragem dinâmica. Pessoas desenvolvedoras podem registrar grandes quantidades de eventos ricos e decidir mais tarde como dividi-los e correlacioná-los. Essa abordagem interativa é útil quando trabalhamos com grandes sistemas distribuídos atuais, porque já passamos do ponto em que podemos antecipar razoavelmente quais perguntas queremos fazer sobre sistemas de produção.

Humio
AVALIE

Humio é um player relativamente novo no espaço de gerenciamento de logs. Desde o início, foi construído para ser extremamente rápido tanto na ingestão de log quanto na de consulta, usando sua linguagem de consulta interna em cima de um banco de dados de séries temporais customizado. Ele se integra com quase tudo, sob a perspectiva de ingestão, visualização e alerta. O espaço de gerenciamento de log tem sido dominado pelo Splunk e pela stack ELK, então, ter alternativas é algo bom. Vamos acompanhar de perto como o Humio se desenvolve.

Kubernetes Operators
AVALIE

Tem nos animado o impacto que o Kubernetes vem tendo em nossa indústria, mas nos preocupa a complexidade operacional que o acompanha. Manter um cluster Kubernetes funcionando e gerenciar pacotes implantados nele requer habilidades especiais e tempo. Processos operacionais como atualizações, migrações, backups, entre outros, podem demandar muito tempo de trabalho. Achamos que Kubernetes Operators terá um papel fundamental na redução dessa complexidade. O framework fornece um mecanismo padrão para descrever processos operacionais automatizados para pacotes em execução em um cluster Kubernetes. Embora o Operators seja liderado e promovido pela RedHat, vários Operators estão sendo desenvolvidos pela comunidade para pacotes comuns de código aberto, como Jaeger, MongoDB e Redis.

OpenAPM
AVALIE

Um dos desafios de se adotar uma alternativa de código aberto para pacotes comerciais populares é, no complicado cenário de projetos, entender quais componentes você precisa, quais funcionam bem juntos e qual é a parte exata que cada componente cobre do todo. Isso é particularmente difícil no mundo da observabilidade, no qual a prática padrão é comprar um pacote abrangente, mas caro, para fazer tudo. A OpenAPM torna mais fácil o processo de seleção ferramentas de observabilidade com código aberto. Ela mostra a safra atual de ferramentas de código aberto, classificadas por suas funções, para que você possa selecionar interativamente ferramentas compatíveis. Desde que você mantenha a ferramenta atualizada, ela deve te ajudar a navegar pelo confuso conjunto de possíveis ferramentas.

Systems
AVALIE

É fácil pensar em vários processos com os quais trabalhamos como cadeias lineares de causa e efeito. Na maior parte do tempo, estamos trabalhando com sistemas mais complexos, em que loops de feedback positivos e negativos influenciam os resultados. O Systems é um conjunto de ferramentas para descrever, executar e visualizar diagramas de sistemas. Usando uma DSL compacta e rodando tanto de forma independente como dentro de um notebook Jupyter, é bem fácil descrever processos bastante complexos e o fluxo de informação por meio deles. É praticamente uma ferramenta de nicho, mas interessante e divertida.

FERRAMENTAS

Humio é um player relativamente novo no espaço de gerenciamento de logs. Desde o início, foi construído para ser extremamente rápido tanto na ingestão de log quanto na de consulta, usando sua linguagem de consulta interna em cima de um banco de dados de séries temporais customizado.

(Humio)

Sistemas um conjunto de ferramentas para descrever, executar e visualizar diagramas de sistemas complexos, nos quais os ciclos de feedback positivos e negativos influenciam os resultados.

(Systems)

FERRAMENTAS

O provedor Terraform para GoCD permite que você construa pipelines usando o Terraform, uma ferramenta madura e amplamente usada no espaço de infraestrutura como de código. Ele possui testes de regressão automática para a API do GoCD, o que deve minimizar os problemas durante a atualização.

(Provedor Terraform para GoCD)

Taurus

AVALIE

Taurus é uma útil ferramenta de testes de performance de serviços e aplicações escrita em Python. Ela engloba muitos executores de testes de performance, incluindo Gatling e Locust. Você pode executá-la da linha de comando e facilmente integrá-la com pipelines de entrega contínua para fazer testes de performance em diferentes estágios da pipeline. Taurus também tem ótimo relatórios, tanto como saída de console baseado em texto quanto integrada com uma UI para web interativa. Nossos times descobriram que configurar arquivos YAML da Taurus é fácil porque você pode usar múltiplos arquivos para descrever cada cenário de teste e consultar as definições de cenário do executor subjacente.

Provedor Terraform para GoCD

AVALIE

O provedor Terraform para GoCD permite que você construa pipelines usando Terraform, uma ferramenta madura e amplamente usada no espaço de infraestrutura como código. Com esse provedor, você pode escrever pipelines na Linguagem de Configuração HashiCorp (HCL), que usa toda a funcionalidade fornecida por Terraform, incluindo áreas de trabalho, módulos e estado remoto. Essa abordagem é uma excelente alternativa ao Gomatic, que destacamos anteriormente no blip sobre pipelines como código. A Golang SDK usada nesse provedor tem testes de regressão automáticos para a API do GoCD, o que deve minimizar os problemas na atualização.

Terratest

AVALIE

Usamos amplamente Terraform como código para configurar infraestruturas de nuvem. Terratest é uma biblioteca Golang que torna mais fácil escrever testes automatizados para código de infraestrutura. Uma execução de teste cria componentes reais de infraestrutura (como servidores, firewalls ou balanceadores de carga), implanta aplicações neles e valida o comportamento esperado usando Terratest. Ao final do teste, Terratest pode "desimplantar" os aplicativos e colocar os recursos em ordem. Isso a torna muito útil para testes ponta a ponta de nossas infraestruturas em um ambiente real.

CloudFormation escrito à mão

EVITE

AWS CloudFormation é uma linguagem declarativa proprietária para provisionar a infraestrutura AWS como código. Arquivos CloudFormation escritos à mão são frequentemente uma abordagem padrão para bootstrap na automação da infraestrutura AWS. Embora isso possa ser uma maneira sensata de se começar um projeto pequeno, nossos times – e a maioria da indústria – acham que CloudFormation escrito à mão simplesmente não escala à medida que a infraestrutura cresce. Armadilhas visíveis de arquivos CloudFormation escritos à mão para projetos grandes incluem legibilidade ruim, falta de construções imperativas, definição e uso de parâmetros limitados e falta de verificação de tipagem. Resolver essas armadilhas levou a um rico ecossistema de ferramentas de código aberto e customizadas. Achamos que Terraform é um padrão sensato que não apenas resolve as armadilhas da CloudFormation, mas também tem uma comunidade ativa que adiciona os recursos mais recentes de AWS e soluciona os problemas. Além da Terraform, você pode escolher dentre muitas outras ferramentas e linguagens, incluindo troposphere, sceptre, Stack Deployment Tool e Pulumi.

LINGUAGENS & FRAMEWORKS

Apollo
ADOTE

Nossos times relatam que Apollo se tornou a biblioteca escolhida para a construção de aplicações React que usam GraphQL para acessar dados em um serviço backend. Embora o projeto da Apollo também forneça um framework de servidor e um gateway GraphQL, o cliente Apollo ganha nossa atenção porque simplifica o problema de conectar componentes de UI a dados providos por um backend GraphQL. De maneira simples, isso significa que menos código precisa ser escrito em comparação com quando se usa backends REST e redux.

MockK
ADOTE

MockK é nossa ferramenta para mocking quando escrevemos testes para aplicações Kotlin. Gostamos de usar essa biblioteca por causa de seu suporte de primeira classe para recursos de linguagem Kotlin, como corrotinas ou blocos lambda. Como uma biblioteca nativa, ela ajuda nossos times a escrever código limpo e conciso ao testar aplicações Kotlin, em vez de usar empacotadores incômodos do Mockito ou do PowerMock.

TypeScript
ADOTE

TypeScript, uma linguagem estaticamente tipada e um superconjunto de JavaScript, tornou-se nosso padrão. Projetos de larga escala são os que mais se beneficiam da segurança de tipagem. Nossas pessoas desenvolvedoras preferem seu gerenciamento mínimo de configuração, o suporte a IDE bem integrado e sua capacidade de refatorar o código com segurança e gradualmente adotar tipos. Com seu bom repositório de definições de tipo TypeScript à mão, nós nos beneficiamos de todas as bibliotecas ricas de JavaScript, ao mesmo tempo em que ganhamos segurança de tipagem.

Apache Beam
EXPERIMENTE

Apache Beam é um modelo de programação unificado de código aberto para definir e executar, em paralelo, pipelines para processamento em lote e streaming de dados. O modelo Beam é baseado no modelo Dataflow, que permite expressar lógica de maneira elegante, para que possamos alternar facilmente entre lote, lote com janelas ou streaming. O ecossistema de processamento de big data evoluiu muito, o que pode tornar difícil escolher o mecanismo correto para processamento de dados. Uma das principais razões para escolher o Beam é que ele permite alternância entre diferentes runners – há alguns meses, o Apache Samza foi adicionado a outros runners já suportados, que incluem Apache Spark, Apache Flink e Google Cloud Dataflow. Runners diferentes têm capacidades diferentes e fornecer uma API portátil é uma tarefa difícil. O Beam tenta encontrar

ADOTE
68. Apollo
69. MockK
70. TypeScript

EXPERIMENTE
71. Apache Beam
72. Formik
73. HiveRunner
74. joi
75. Ktor
76. Laconia
77. Puppeteer
78. Reactor
79. Resilience4j
80. Room
81. Rust
82. WebFlux

AVALIE
83. Aeron
84. Arrow
85. Chaos Toolkit
86. Dask
87. Embark
88. fastai
89. http4k
90. Immer
91. Karate
92. Micronaut
93. Next.js
94. Pose
95. react-testing-library
96. ReasonML
97. Taiko
98. Vapor

EVITE

LINGUAGENS & FRAMEWORKS

joi é uma ferramenta para a descrição de esquemas e validação de objetos em JavaScript, independente de qualquer framework de aplicação web.

(joi)

Sistemas reativos trazem escalabilidade melhorada e resiliência, mas com custo aumentado de depuração e uma curva de aprendizado mais acentuada. Alguns de nossos projetos observaram melhoras significativas em escalabilidade uma vez que foram movidos para Reactor e o resto da stack reativa.

(Reactor)

um equilíbrio delicado ao colocar as inovações desses runners dentro do modelo Beam e também trabalhar com a comunidade para influenciar o roadmap desses runners. O Beam tem SDKs em múltiplas linguagens, incluindo Java, Python e Golang. Também tivemos sucesso usando Scio, que fornece um empacotador Scala em torno do Beam.

Formik
EXPERIMENTE

Formik é um útil componente de ordem superior para tornar o trabalho complexo e detalhado de lidar com formulários em React muito mais fácil. Ele localiza o gerenciamento de estado, ajuda com a submissão e, opcionalmente, usa Yup para simplificar a validação de dados.

HiveRunner
EXPERIMENTE

HiveRunner é um framework de testes unitários de código aberto para as queries do Apache Hadoop Hive baseado em JUnit4. Ao escrever analíticas não-triviais ou pipelines de dados em Hive SQL, descobrimos que o HiveRunner pode ser um bom facilitador para escrever testes e mesmo aplicar TDD a alguns SQL moderadamente complicados. O HiveRunner permite que você escreva Hive SQL como artefatos testados e implantáveis.

joi
EXPERIMENTE

joi é uma ferramenta para a descrição de esquemas e validação de objetos em JavaScript. Gostamos do fato de joi ser independente de qualquer framework de aplicação web, pois nossos times podem usar os mesmos esquemas em diferentes stacks. Também é possível usar bibliotecas complementares para gerar documentação Swagger para APIs que validam solicitações com esquemas joi.

Ktor
EXPERIMENTE

Kotlin mostrou seu valor para além do desenvolvimento de aplicativos para dispositivos móveis. Ao construir microsserviços e enviar o software para produção, nossos times têm tido boas experiências com o Ktor. Ktor é um framework que, diferentemente de outros frameworks web que suportam Kotlin, é escrito em Kotlin, usando recursos da linguagem como corrotinas, que permitem uma implementação assíncrona sem bloqueios. A flexibilidade de incorporar diferentes ferramentas para criação de logs, injeção de dependência (DI) e um mecanismo de templates – além de sua arquitetura leve – tornam Ktor uma opção interessante para nossos times na criação de serviços RESTful.

Laconia
EXPERIMENTE

Laconia é um framework para desenvolver funções AWS Lambda em JavaScript. À medida que o interesse e o uso de tecnologias sem servidor cresceram, também cresceu a complexidade das aplicações construídas. Laconia é um framework leve e pequeno que apara algumas das arestas que muitas vezes encontramos. Ele usa injeção de dependência para isolar o código da sua aplicação de APIs AWS de mais baixo nível e fornece adaptadores para diferentes eventos aos quais sua aplicação pode responder também. Ele também funciona bem com o framework Serverless no momento da implantação. Gostamos de frameworks simples e pequenos, e o Laconia é tudo isso.

Puppeteer
EXPERIMENTE

A exemplo de Cypress e TestCafe, Puppeteer é uma das ferramentas de teste de UI para web que recebe elogios de nossos times. Puppeteer pode ter controle fino sobre navegadores headless, obter rastreamento de tempo para diagnósticos de performance e muito mais. Nossos times acharam Puppeteer mais rápida e mais flexível que outras alternativas baseadas no WebDriver.

Reactor
EXPERIMENTE

Já falamos sobre Reactor em edições anteriores do Radar, e ele continua a ganhar popularidade em muitos de nossos projetos. Com o ecossistema Spring suportando Reactor, a biblioteca se tornou a implementação principal de Reactive Streams. Sistemas reativos trazem escalabilidade melhorada e resiliência, mas com custo aumentado de depuração e uma curva de aprendizado mais acentuada. Para projetos em que essa troca é aceitável, Reactor tem provado ser uma boa escolha. Alguns de nossos projetos observaram melhoras significativas em escalabilidade uma vez que foram movidos para Reactor e o resto da stack reativa. Com R2DBC, estamos começando a ter suporte reativo para drivers RDBMS, que resolve uma das fraquezas dos serviços reativos.

Resilience4j
EXPERIMENTE

Resilience4j é uma biblioteca leve de tolerância a falhas inspirada na Hystrix, da Netflix. Gostamos de sua estrutura leve e modular, na qual usamos módulos específicos para recursos específicos, como circuit braker, limitação de taxa, retry e bulkhead. Enquanto malhas de serviço estão assumindo alguns recursos de tolerância a falhas, as bibliotecas de tolerância a falhas continuam a ser um componente-chave de nossos sistemas para um comportamento de tolerância a falhas específico de domínio e para serviços sem contêiner. Com Hystrix entrando em modo de manutenção, a Resilience4j se torna a escolha padrão no ambiente Java. Ela pode operar tanto com APIs sincrônicas como com as reativas. E também fornece métricas para dropwizard metrics, Prometheus, etc., usando módulos adicionais.

Room
EXPERIMENTE

Room é uma biblioteca de persistência para acessar o SQLite no Android. Ela torna o código de acesso a dados muito mais simples, com código boilerplate mínimo e mais robusto, com verificação de tempo de compilação de queries SQL. Nossas pessoas desenvolvedoras gostam de sua integração completa com queries observáveis, usando o LiveData. Room é um dos componentes do Jetpack para Android que foram criados para tornar o desenvolvimento de aplicações em Android mais fácil.

Rust
EXPERIMENTE

Desde que a abordamos no Radar de janeiro de 2015, temos visto um interesse crescente em Rust. Alguns de nossos clientes estão usando Rust atualmente, em grande parte no contexto de ferramentas de infraestrutura, mas também em dispositivos internos de alta potência. O interesse foi alimentado por um ecossistema em crescimento, assim como melhorias na linguagem em si, que incluem melhorias diretas de performance, mas também mudanças que deixaram Rust mais intuitiva, como, por exemplo, a mudança para um escopo não-léxico. A maioria das mudanças significativas estão na Rust 2018 standard, lançada em dezembro do ano passado.

WebFlux
EXPERIMENTE

WebFlux é a implementação Spring Framework do Reactive Streams. Vemos um crescimento em modelos de programação reativa em nossos times em geral e o uso de WebFlux em times que estão trabalhando no ecossistema Spring. É mais bem usada em grandes ecossistemas de microsserviços, em que a alta performances das solicitações é uma preocupação. Ela permite sobrepor o processamento de solicitações de maneira assíncrona sem as complicações de se usar múltiplas threads. WebFlux usa Reactor como sua biblioteca reativa, mas é interoperável com outras bibliotecas reativas via Reactive Streams. Usa Netty como seu mecanismo fundamental de comunicações de alta performance. Embora incentivemos o uso do Reactive Streams, adotar esse modelo de programação requer uma significativa mudança de pensamento.

Aeron
AVALIE

Aeron é um transporte de mensagens ponto a ponto eficiente e confiável. Ele fornece um log persistente e replicado de mensagens por meio de vários drivers de mídia, incluindo HTTP, UDP e TCP. Também suporta armazenamento persistente de fluxos de mensagens para reprodução posterior. Para muitas aplicações, o Aeron pode ser um exagero porque opera em um nível muito baixo (OSI Camada 4, conceitualmente), mas seu design ponto a ponto e latência baixa (e previsível) são úteis para vários usos. Na verdade, achamos que é útil em certas aplicações de aprendizado de máquina, assim como fazendo parte de arquiteturas baseadas em eventos. Achamos válido apontar que existem protocolos alternativos de mensagens que não demandam serviços adicionais, como Apache Kafka, para serem executados.

Arrow
AVALIE

Arrow é uma biblioteca de programação funcional para Kotlin, criada pela fusão de duas bibliotecas populares já existentes (kategory e funKTionale). Enquanto

LINGUAGENS & FRAMEWORKS

Room torna o acesso à base de dados Android mais simples, com código boilerplate mínimo e mais robusto e verificação de tempo de compilação de queries SQL.

(Room)

Dask fornece maneiras de escalar fluxos de trabalho pandas, Scikit-Learn e Numpy com reescrita mínima. Isso o torna uma escolha interessante para cientistas de dados que procuram escalabilidade horizontal para grandes conjuntos de dados.

(Dask)

LINGUAGENS & FRAMEWORKS

fastai é uma biblioteca Python de código aberto que simplifica o treinamento de redes neurais rápidas e precisas e possui suporte nativo para visão computacional, processamento de linguagem natural (NLP) e muito mais.

(fastai)

Kotlin fornece construções básicas para programação funcional, Arrow entrega um pacote de abstrações de alto nível prontas para o uso de pessoas desenvolvedoras. Ela fornece tipos de dados, classes de tipos, effects, optics e outros padrões de programação funcional, assim como integração com bibliotecas populares. Com Arrow, as bibliotecas existentes são unificadas, o que deve ajudar a evitar comunidades divididas neste espaço.

Chaos Toolkit

AVALIE

O Chaos Toolkit é uma entre as várias ferramentas da Engenharia do Caos que entraram nesta edição do Radar. O kit é usado para descrever e então rodar experimentos repetíveis em sua infraestrutura, para entender sua resiliência em caso de falha. Muitos de nossos times estão usando ferramentas domésticas para fazer isso, então, é ótimo ver um projeto de código aberto dedicado a essa prática. O conjunto de ferramentas já tem drivers para AWS, Azure Service Fabric e GCE (entre outros) e funciona bem com ferramentas de compilação, o que permite experimentar com automação. Porém, as ressalvas usuais se aplicam – a Engenharia do Caos é uma técnica poderosa que é mais bem usada em sistemas com capacidade de resiliência, ou seja, sistemas que foram construídos para lidar com falhas. Por esta razão, recomendamos começar a usar o Chaos Toolkit em ambientes que não sejam de produção.

Dask

AVALIE

Cientistas e pessoas engenheiras de dados frequentemente usam bibliotecas, como pandas, para fazer uma análise ad hoc. Embora expressivas e poderosas, essas bibliotecas têm uma limitação importante: elas funcionam apenas em um único CPU e não fornecem escalabilidade horizontal para grandes grupos de dados. Dask, entretanto, inclui um agendador leve e de alta performance que pode escalar de um laptop a um cluster de máquinas. E por trabalhar com NumPy, pandas e Scikit-learn, Dask parece promissora para uma avaliação mais aprofundada.

Embark

AVALIE

Recomendamos Truffle para o desenvolvimento de aplicações descentralizadas anteriormente. Embark também pode tornar seu trabalho mais fácil. Embark fornece funcionalidades como scaffolding, compilação, testes e depuração, além de integrar com armazenamentos descentralizados como IPFS. Por meio de sua configuração declarativa, você pode gerenciar configuração, dependências, artefatos e implantação de contratos inteligentes facilmente. O painel CLI interativo da Embark também é impressionante. Vemos pessoas usando Remix para escrever contratos inteligentes e fazer a implantação manual de seus aplicativos sem testes automatizados, gerenciamento de controle de versão ou gerenciamento de artefato. Gostaríamos de chamar a atenção para a prática da engenharia de aplicações descentralizadas, promovendo ferramentas como Truffle e Embark.

fastai

AVALIE

fastai é uma biblioteca Python de código aberto que simplifica o treinamento de redes neurais rápidas e precisas. É construído sobre o PyTorch e se tornou uma ferramenta popular entre nossas cientistas de dados. fastai simplifica aspectos dolorosos do treinamento de modelos, como pré-processamento e carregamento de dados para algumas linhas de código. É construída com boas práticas de aprendizagem profunda e tem suporte nativo para visão computacional, processamento de linguagem natural (NLP) e muito mais. A motivação do grupo de fundadores foi criar uma biblioteca fácil de usar para aprendizado profundo e um sucessor melhorado do Keras. GCP, AWS e Azure rapidamente incluíram o fastai em suas imagens de máquina. Os criadores do fastai, reconhecendo as limitações de velocidade e segurança de Python, anunciaram a inclusão do Swift como uma linguagem alternativa para aprendizado profundo. Observaremos esse espaço de perto.

http4k
AVALIE

http4k é um kit de ferramentas HTTP escrito em Kotlin puro, para fornecer e consumir serviços em HTTP. Uma das ideias fundamentais por trás do http4k é que os aplicativos em HTTP são modelados compondo-se duas simples funções – HttpHandler e Filter. Elas foram inspiradas no artigo "Your Server as a Function", do Twitter. É muito leve e o módulo principal não tem dependências além do Kotlin StdLib. Além de sua elegância e simplicidade, também gostamos de sua ênfase em testes — dado que as entidades nas bibliotecas são imutáveis e as rotas no aplicativo, assim como o aplicativo em si, são apenas funções, elas são extremamente fáceis de testar. Uma das coisas para se atentar, contudo, é que ainda não há suporte para corrotinas ou sem bloqueios no http4k.

Immer
AVALIE

Com a crescente complexidade de aplicações JavaScript de página única, gerenciar a previsibilidade do estado está se tornando cada vez mais importante. A imutabilidade pode ajudar a assegurar que nossas aplicações se comportem consistentemente, mas infelizmente o JavaScript não suporta nativamente a capacidade de criar objetos imutáveis. Bibliotecas como Immutable.js preenchem essa lacuna, mas introduzem novos problemas com a existência de dois tipos de objetos e arranjos na aplicação, a biblioteca JavaScript nativa e a versão. Immer — "sempre" em alemão — é um pacote pequeno que permite trabalhar com estado imutável de maneira mais conveniente. É baseado no mecanismo cópia-em-gravação, tem uma API mínima e opera em objetos e arranjos normais em JavaScript. Isso significa que o acesso a dados ocorre sem obstáculos e não são necessários grandes esforços de refatoração ao introduzir imutabilidade a um código existente.

Karate
AVALIE

Dada nossa experiência de que testes são as únicas especificações de API que realmente importam, estamos sempre à procura de novas ferramentas que possam nos ajudar. Karate é um framework de testes de API cujo recurso único é que os testes são escritos diretamente em Gherkin, sem depender de uma linguagem de programação de uso geral para implementar o comportamento de teste. Karate é, na verdade, uma linguagem específica de domínio para descrever testes de API baseada em HTTP. Embora essa abordagem seja interessante e sirva para algumas especificações legíveis para testes simples, a linguagem com fins especiais para corresponder e validar cargas úteis podem se tornar cheia de sintaxe e difíceis de entender. Resta ver se testes mais complexos escritos neste estilo serão legíveis e sustentáveis no longo prazo.

Micronaut
AVALIE

Micronaut é um novo framework que utiliza JVM para construir microsserviços usando Java, Kotlin ou Groovy. Destaca-se por um pequeno uso de memória e um curto tempo de inicialização. Ele consegue essas melhorias evitando o uso de reflections em runtime para geração de injeção de dependências (ID) e proxy, uma falha comum dos frameworks tradicionais. Em vez disso, usa um contêiner ID/AOP que faz a injeção de dependências na hora da compilação. Isso o torna atrativo não apenas pelos microsserviços padrão para web, mas também no contexto da Internet das Coisas, aplicações para Android e funções sem servidor, por exemplo. O Micronaut usa Netty e tem suporte de primeira classe para programação reativa. Também inclui muitos recursos que fazem dele cloud-native friendly, como descoberta de serviço e circuit breaker. O Micronaut é um estreante muito promissor para o framework full stack no espaço JVM e estamos otimistas para observá-lo.

Next.js
AVALIE

O React.js revolucionou a forma como a maioria das pessoas escreve aplicações em JavaScript de página única. Geralmente, recomendamos o uso do app Create React durante o ciclo de vida da aplicação, para que você não tenha que configurar manualmente sua

LINGUAGENS & FRAMEWORKS

O http4k é um kit de ferramentas HTTP escrito em Kotlin puro, para fornecer e consumir serviços em HTTP. É elegante e simples, com ênfase na testabilidade.

(http4k)

Micronaut é um novo framework que utiliza JVM para construir microsserviços usando Java, Kotlin ou Groovy. Destaca-se por um pequeno uso de memória e um curto tempo de inicialização.

(Micronaut)

LINGUAGENS & FRAMEWORKS

O Taiko é uma biblioteca node.js com uma API limpa e concisa para auxiliar a automação em navegadores chrome ou chromium. Testes escritos em Taiko tem como objetivo serem altamente legíveis e sustentáveis.

(Taiko)

configuração, construção e pacotes. Mas algumas pessoas desenvolvedoras vão preferir uma ferramenta cujos padrões iniciais refletem um conjunto de opiniões saudáveis. O Next.js é esse framework, e tem atraído o interesse de entusiastas de frontend. O Next.js simplifica o roteamento, renderiza no servidor por padrão e agiliza dependências e construções. Estamos otimistas para ver se ele atenderá as expectativas em nossos próprios projetos.

Pose
AVALIE

Pose é uma biblioteca de animação simples do tipo CSS para frameworks React.js, React Native e Vue.js. É um sistema de movimento declarativo que combina a simplicidade da sintaxe do CSS com a força e a flexibilidade das animações e interações em JavaScript.

react-testing-library
AVALIE

Desde que o ritmo dos frameworks em JavaScript diminuiu, nossos times têm mais tempo para trabalhar com frameworks específicos e estão obtendo insights mais profundos como resultado disso. Com React e o framework de testes dominante, Enzyme, observamos uma tendência preocupante de testes de unidade tornando-se intimamente acoplados a detalhes de implementação sem fornecer — por causa do foco em detalhes simples — muita confiança de que os recursos funcionam como o esperado. Esses testes de unidade tornam a evolução do design difícil e transferem muita responsabilidade da pirâmide de testes para testes funcionais. Isso nos fez revisitar a ideia de testes subcutâneos. Além disso, por causa de seu design, o Enzyme apresenta problemas ao tentar acompanhar o desenvolvimento do React. Tudo isso nos levou a considerar react-testing-library como um novo framework para testar aplicações em React.

ReasonML
AVALIE

ReasonML é uma nova e interessante linguagem baseada em OCaml com toques de sintaxe do tipo C e JavaScript como alvo de compilação padrão. Criada pelo Facebook, ela permite incorporar fragmentos de JavaScript e template JSX com boa integração com React. Visa ser acessível para pessoas desenvolvedoras de JavaScript e impulsiona esse ecossistema, ao mesmo tempo em que fornece segurança de tipagem em uma linguagem funcional.

Taiko
AVALIE

Taiko é uma biblioteca node.js com uma API limpa e concisa para auxiliar a automação em navegadores chrome ou chromium. Você pode aproveitar os seletores inteligentes da Taiko e escrever testes confiáveis à medida que a estrutura da aplicação web evolui. Não há necessidade de seletores id, CSS ou XPath ou adicionar esperas explícitas (para solicitações XHR) em scripts de teste. O gravador interativo REPL é útil quando você quer desenvolver os testes lado a lado enquanto explora a funcionalidade. Embora você possa usar Taiko de maneira independente, tivemos muito sucesso usando a biblioteca com Gauge.

Vapor
AVALIE

Somos fortes proponentes da programação poliglota, mas reconhecemos que em alguns casos pode fazer sentido focar em uma única linguagem de programação. Se você tem um envolvimento profundo com Swift (provavelmente devido a desenvolvimento para iOS) e está à procura de uma tecnologia para escrever serviços com servidor web, dê uma olhada em Vapor , um moderno framework web para Swift que ganhou uma boa popularidade.

Quer se atualizar com artigos e informações relacionadas ao radar?
Siga a gente nas redes sociais ou torne-se assinante.

assine

ThoughtWorks®

Somos uma consultoria global de software e uma comunidade de pessoas apaixonadas e guiadas por propósitos. Pensamos de maneira disruptiva para entregar tecnologia capaz de enfrentar os desafios mais difíceis de nossas clientes, ao mesmo tempo em que buscamos revolucionar a indústria de TI e provocar uma mudança social positiva.

Fundada há 25 anos, a ThoughtWorks se tornou uma empresa com mais de 6000 pessoas, incluindo uma área de produtos que desenvolve ferramentas pioneiras para times de software. A ThoughtWorks tem 40 escritórios em 14 países: Alemanha, Austrália, Brasil, Canadá, Chile, China, Equador, Espanha, Estados Unidos, Índia, Itália, Reino Unido, Singapura e Tailândia.

thoughtworks.com

ThoughtWorks®

thoughtworks.com/radar

#TWTechRadar