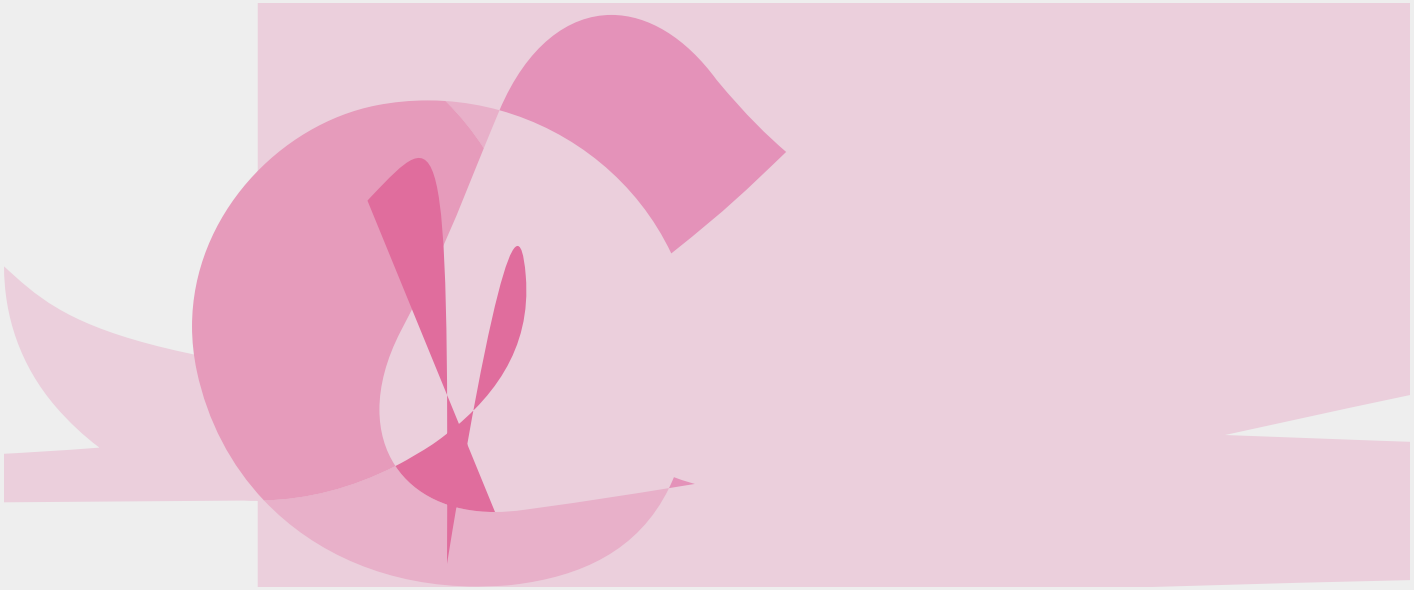
ThoughtWorks®

TECHNOLOGY RADAR VOL.19

Nossas ideias sobre tecnologias e tendências que estão moldando o futuro



CONTRIBUIÇÕES

O Technology Radar é produzido pelo Conselho Consultivo de Tecnologia da ThoughtWorks, composto por:

contidos no Contexto Delimitado de um único recurso do negócio. Essa modularidade e capacidade de entrega independente deve ser incluída nos critérios de aceitação de um processo de escolha de fornecedores.

Manter o controle adequado sobre dados confidenciais é difícil, especialmente quando – para fins de backup e recuperação – os dados são copiados fora de um sistema mestre de registro. A **DESTRUIÇÃO CRIPTOGRAFADA** é a prática de tornar os dados confidenciais ilegíveis, sobrescrevendo ou excluindo deliberadamente as chaves de criptografia usadas para proteger esses dados. Por exemplo, uma tabela inteira de detalhes pessoais do cliente pode ser criptografada usando chaves aleatórias para cada registro, com uma tabela diferente armazenando as chaves. Se um cliente exercitou seu "direito de ser esquecido", podemos simplesmente excluir a chave apropriada, efetivamente "destruindo" os dados criptografados. Essa técnica pode ser útil quando temos confiança em manter o controle adequado de um conjunto menor de chaves de criptografia, mas temos menos confiança em relação ao controle sobre um conjunto de dados maior.

O relatório State of DevOps e o livro subsequente Accelerate destacam um fato surpreendente: para prever e melhorar o desempenho de um time, precisamos apenas medir lead time, frequência de implantação, tempo médio de restauração (MTTR) e alterar a porcentagem de falhas.

(Quatro métricas fundamentais)

O relatório State of DevOps, publicado pela primeira vez em 2014, afirma que times de alto desempenho criam organizações de alto desempenho. Recentemente, o time por trás do relatório publicou Accelerate, que descreve o método científico usado no relatório. Uma das principais conclusões de ambos os materiais são as **QUATRO MÉTRICAS FUNDAMENTAIS** para sustentar o desempenho de entrega de software: lead time, frequência de implantação, tempo médio de restauração (MTTR) e porcentagem de falha de alteração. Como uma consultoria que ajudou na transformação de muitas organizações, vemos essas métricas surgirem frequentemente como uma forma de ajudar as organizações a determinar se estão melhorando o desempenho geral. Cada métrica cria um ciclo virtuoso e direciona o foco dos times à melhoria contínua: para reduzir o lead time, você reduz as atividades desnecessárias que, por sua vez, permitem implantar com mais frequência; a frequência de implantação força seus times a melhorar suas práticas e automação; sua velocidade para se recuperar de falhas é aprimorada por melhores práticas, automação e monitoramento, que reduzem a frequência de falhas.

O autoatendimento sob demanda é uma característica fundamental (e benéfica) da computação em nuvem. Quando cenários de serviços de larga escala são implantados usando uma única conta, regras e processos relacionados ao uso dessa conta se tornam necessários, geralmente envolvendo etapas de aprovação que aumentam o tempo de resposta. Uma abordagem melhor é a de **CONFIGURAÇÃO DE NUVEM PARA MÚLTIPLAS CONTAS**, em que várias contas são usadas; em casos exgtremos uma conta por equipe. Isso de fato adiciona custos em outros lugares, por exemplo, para garantir o faturamento compartilhado, permitir a comunicação entre VPCs e gerenciar o relacionamento com o provedor de nuvem. No entanto, isso geralmente acelera o desenvolvimento e melhora a segurança, porque as contas de serviço único são mais fáceis de auditar e, no caso de uma violação, o impacto é bastante reduzido. Ter várias contas também reduz a dependência de provedor de novem, porque uma conta fornece um bom limite para serviços que podem ser movidos em bloco para outro provedor de nuvem.

A observabilidade é parte integrante da operação de uma arquitetura de microsserviços distribuída. Dependemos de diferentes saídas do sistema, como rastreamento distribuído, registros agregados e métricas para inferir o estado interno dos componentes distribuídos, diagnosticar onde estão os problemas e chegar à raiz da causa. Um aspecto importante de um ecossistema de observabilidade é o monitoramento – visualização e análise da saída do sistema – e o alerta quando condições inesperadas são detectadas. Tradicionalmente, a configuração de painéis de monitoramento e a configuração de alertas são feita por meio de sistemas de apontar e clicar baseados em GUI. Essa abordagem leva a configurações de painel não repetíveis, incapacidade de testar e ajustar continuamente os alertas para evitar a fadiga ou a perda de alertas importantes, além do distanciamento de boas práticas das organizações. É altamente recomendável tratar suas configurações do ecossistema de **OBSERVABILIDADE COMO CÓDIGO**, além de adotar infraestrutura como código para sua infraestrutura de monitoramento e alertas. Escolha produtos de observabilidade que suportam configuração

por meio de código controlado por versão e execução de APIs, ou comandos por meio de pipelines de CD de infraestrutura. A observabilidade como código é um aspecto muitas vezes esquecido da infraestrutura como código, e acreditamos que seja crucial o suficiente para ser destacada.

A observabilidade é parte integrante da operação de uma arquitetura distribuída e baseada em microsserviços. Recomendamos tratar suas configurações de ecossistema de observação como código..

(Observabilidade como código)

Muitas vezes, em um esforço para terceirizar o risco para seus fornecedores, as empresas buscam um único fornecedor para suas implementações de sistema mais críticas e arriscadas. Infelizmente, isso proporciona

é extremamente valioso. Para obter uma análise reproduzível, tanto os dados quanto o modelo (incluindo escolha de algoritmo, parâmetros e hiperparâmetros) precisam ser versionados. Os **DADOS VERSIONADOS PARA ANÁLISES REPRODUZÍVEIS** são um problema relativamente mais complicado do que os modelos de versão, devido ao tamanho dos dados. Ferramentas como DVC ajudam na criação de versões de dados ao permitir que os usuários executem commit e push de arquivos de dados para um repositório de armazenamento remoto em nuvem, usando um fluxo de trabalho parecido com o do git. Isso facilita a obtenção de uma versão específica de dados para reproduzir uma análise.

KATAS DE CAOS é uma técnica que nossos times desenvolveram para treinar e aperfeiçoar pessoas engenheiras de infraestrutura e de plataforma. Eles combinam técnicas de Engenharia de Caos – que é a criação de falhas e interrupções em um ambiente controlado – com a abordagem sistemática de ensino e treinamento Kata. Aqui, Kata refere-se a padrões de código que acionam falhas controladas, permitindo que as pessoas engenheiras descubram o problema, recuperem-se da falha, realizem um postmortem e encontrem a causa raiz. A execução repetida dos Katas ajuda as pessoas engenheiras a internalizar suas novas habilidades.

Ao criar imagens do Docker para nossas aplicações, geralmente temos duas preocupações: a segurança e o tamanho da imagem. Tradicionalmente, usamos ferramentas de varredura de segurança de contêiner para detectar e corrigir vulnerabilidades e exposições comuns e pequenas distribuições como Alpine Linux, para endereçar o tamanho da imagem e o desempenho da distribuição. Neste edição do Radar, estamos felizes em abordar a segurança e o tamanho dos contêineres com uma nova técnica chamada **IMAGENS DOCKER SEM DISTRIBUIÇÃO**, desenvolvida pelo Google. Com essa técnica, o footprint da imagem é reduzido para a aplicação, seus recursos e dependências de tempo de execução de linguagem, sem a distribuição do sistema operacional. As vantagens dessa técnica incluem redução do ruído dos scanners de segurança, menor superfície de ataque à segurança, sobrecarga reduzida de vulnerabilidades de patch e tamanho de imagem ainda menor para melhor desempenho. O Google publicou um conjunto de imagens de contêiner sem distribuição para diferentes linguagens. Você pode criar imagens de aplicativos sem distribuição usando a ferramenta de criação do Google Bazel, que tem regras para criar contêineres sem distribuição ou simplesmente usar Dockerfiles em vários estágios. Observe que os contêineres sem distribuição, por padrão, não possuem um shell para depuração. No entanto, você pode encontrar facilmente versões de depuração de contêineres sem distribuição online, incluindo busybox shell.

Ao criar imagens do Docker para nossas aplicações, muitas vezes nos preocupamos com duas coisas: a segurança e o tamanho da imagem. Com essa técnica, o footprint da imagem é reduzido para o aplicativo, seus recursos e dependências de tempo de execução de linguagem, sem a distribuição do sistema operacional.

(Imagens Docker sem distribuição)

A ThoughtWorks, como pioneira e líder no espaço ágil, tem proposto a prática da entrega incremental.

Watchmen é uma ferramenta interessante, criada para prover garantia orientada por regras para configurações de contas da AWS que são controladas e operadas de forma independente pelos times de desenvolvimento. O Scout2 é outro exemplo desses analisadores para dar suporte à adequação aos princípios de segurança.

Em arquiteturas e implementações mais complexas, pode não ser imediatamente óbvio que uma compilação dependente do código atualmente sendo checked-in está quebrada. As pessoas desenvolvedoras que tentam consertar uma compilação quebrada podem se encontrar trabalhando contra um alvo em movimento, já que a compilação é continuamente acionada por dependências upstream. **VERIFICAÇÕES DE PRÉ-COMMIT DE COMPILAÇÃO DOWNSTREAM** é uma técnica muito simples: um script pré-commit ou pré-push verifica o status dessas compilações downstream e alerta a pessoa desenvolvedora com antecedência que uma compilação está quebrada.

Conforme grandes organizações transicionam para times mais autônomos que são proprietários e operam seus próprios microsserviços, como elas podem assegurar a consistência e compatibilidade necessárias entre esses serviços sem depender de uma infraestrutura de hospedagem centralizada? Para trabalhar em conjunto de forma eficiente, mesmo microsserviços autônomos precisam se alinhar com alguns padrões organizacionais. Uma **MALHA DE SERVIÇOS** oferece descoberta, segurança, rastreamento, monitoramento e processamento de falhas consistentes sem a necessidade de um recurso compartilhado como um API gateway ou um ESB. Uma implementação típica envolve um processo de proxy reverso leve implantado juntamente com cada processo de serviço, possivelmente em um contêiner separado. Esses proxies implantado juntam
se comunicam com registros de serviço, prove8 ( )rrerviço, prove8 ( )rrerviço, prove8 ( )rreo des5m

para manter a independência da nuvem somente quando necessário, e usando Polycloud para misturar e combinar serviços de diferentes fornecedores onde isso faz sentido. Em resumo, mude sua abordagem de um uso genérico da nuvem para uma estratégia sensata de várias nuvens.

Uma das características que define uma arquitetura de microsserviços é que os componentes e serviços do sistema são organizados em torno de capacidades de negócio. Independentemente do tamanho, os microsserviços encapsulam um agrupamento significativo de funcionalidade e informação para permitir a entrega independente de valor de negócio. Isso contrasta com abordagens anteriores na arquitetura de serviços que os organizavam de acordo com características técnicas. Observamos uma série de empresas adotando uma **ARQUITETURA DE MICROSSERVIÇOS EM CAMADAS**, que, em alguns aspectos, é uma contradição. Essas empresas recorreram à organização de serviços principalmente de acordo com uma função técnica, por exemplo, APIs de experiência, APIs de processo ou APIs do sistema. É muito fácil atribuir times de tecnologia por camada, de modo que entregar qualquer alteração de negócio importante requer uma coordenação lenta e cara entre vários times. Nós nos prevenimos contra os efeitos dessa divisão por camadas e recomendamos a organização de serviços e times essencialmente de acordo com capacidades de negócio.

O **GERENCIAMENTO DE DADOS MESTRE** (MDM) é um exemplo clássico da solução empresarial "bala de prata": ela promete resolver uma classe de problemas aparentemente relacionados de uma só vez. Por meio da criação de um único ponto centralizado de mudança, coordenação, teste e implantação, as soluções de MDM afetam negativamente a capacidade de uma organização de responder às mudanças nos negócios. As implementações tendem a ser longas e complexas, pois as organizações tentam capturar e mapear todos os dados "mestre" no MDM, integrando a solução MDM em todos os sistemas consumidores ou produtores.

Os microsserviços surgiram como uma das principais técnicas de arquitetura em sistemas modernos baseados em nuvem, mas ainda acreditamos que os times devem proceder com cuidado ao fazer essa escolha. A **INVEJA DE MICROSSERVIÇOS** tenta os times a complicarem sua arquitetura com muitos serviços, simplesmente porque é uma opção de arquitetura moderna. Plataformas como o Kubernetes facilitam muito a implementação de conjuntos complexos de microsserviços, e os fornecedores estão empurrando suas soluções para gerenciamento de microsserviços, potencialmente influenciando times a seguirem nesse caminho. É importante lembrar que os microsserviços trocam a complexidade de desenvolvimento para complexidade operacional, e exigem uma base sólida de testes automatizados, entrega contínua e cultura de DevOps.

Em várias ocasiões, vimos designs de sistema que usam **EVENTOS DE SOLICITAÇÃO-RESPOSTA EM FLUXOS DE TRABALHO VOLTADOS AO USUÁRIO**. Nesses casos, a interface do usuário é bloqueada ou o usuário precisa aguardar o carregamento de uma nova página até que uma mensagem de resposta correspondente a uma mensagem de solicitação seja recebida. Os principais motivos citados para designs como este são o desempenho ou uma abordagem unificada para comunicação entre back-ends para casos de uso síncronos e assíncronos. Acreditamos que o aumento da complexidade – em desenvolvimento, testes e operações – supera em muito o benefício de ter uma abordagem unificada, e sugerimos fortemente o uso de solicitações HTTP síncronas quando a comunicação síncrona entre os serviços de back-end for necessária. Quando bem implementada, a comunicação usando HTTP raramente é um gargalo em um sistema distribuído.

A automação de processos robóticos (**RPA**) é uma parte essencial de muitas iniciativas de transformação digital, pois promete proporcionar redução de custos sem a necessidade de modernizar a arquitetura e os sistemas subjacentes. O problema com essa abordagem, que se concentrar apenas na automação de processos de negócios, sem abordar os sistemas ou recursos de software subjacentes, é que isso pode dificultar ainda mais a alteração dos sistemas subjacentes, introduzindo acoplamentos adicionais. Isso torna ainda mais difícil qualquer tentativa futura de endereçar o cenário de TI legado. Poucos sistemas podem se dar ao luxo de ignorar mudanças e, portanto, os esforços de RPA precisam ser associados a estratégias apropriadas de modernização de sistemas legados.

plausível, que inclui ofertas exclusivas, como Azure Alemanha e Azure Stack, e que dá alguma certeza às empresas européias se antecipando ao GDPR e possíveis mudanças legislativas nos Estados Unidos.

Sistemas de gerenciamento de conteúdo headless (CMSes) estão se tornando um componente comum de plataformas digitais. **CONTENTFUL** é um CMS headless moderno que nossos times integraram com êxito em seus fluxos de desenvolvimento. Gostamos particularmente de 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 aplicar práticas de design de banco de dados evolucionário ao desenvolvimento do CMS. Outros recursos notáveis que gostamos incluem a adição de duas CDNs por padrão para entregar ativos de mídia e documentos JSON, bom suporte para localização e a capacidade – embora com algum esforço – de integração com Auth0.

Como a **GOOGLE CLOUD PLATFORM** (GCP) cresceu em termos de regiões geográficas disponíveis e maturidade dos serviços, clientes em todo o mundo agora podem considerá-lo seriamente para sua estratégia na nuvem. Em algumas áreas, a GCP atingiu paridade de funcionalidades com sua principal concorrente, Amazon Web Services, enquanto em outras áreas se diferenciou — de forma notável com plataformas de aprendizagem de máquina acessíveis, ferramentas de engenharia de dados e Kubernetes operável como uma solução de serviço (GKE). Na prática, nossos times só têm elogios à experiência de desenvolvimento trabalhando com as ferramentas e APIs da GCP.

Um padrão recomendado é usar uma rede de nuvem privada virtual gerenciada no nível organizacional e dividida em sub-redes menores sob o controle de cada time de entrega. A VPC compartilhado torna organizações, projetos, VPCs e sub-redes entidades de primeira classe em confgurações de rede – isso simplifca a confguração e torna a segurança e o controle de acesso mais transparentes.

(VPC compartilhada)

À medida que ganhamos mais experiência com a nuvem pública em organizações grandes e pequenas, surgiram alguns padrões. Um deles é uma rede de nuvem privada virtual gerenciada no nível organizacional e dividida em sub-redes menores sob o controle de cada time de entrega. Isso está intimamente relacionado à idéia de configuração de múltiplas contas na nuvem e ajuda a partição de infraestrutura alinhada com limites do time. Depois de realizar muitas vezes essa configuração explicitamente usando VPCs, sub-redes, grupos de segurança e NACLs, gostamos muito da noção de **VPC COMPARTILHADA** do Google. A VPC compartilhada transforma organizações, projetos e sub-redes em entidades de primeira classe em configurações de rede. As VPCs podem ser gerenciadas por administradores de uma organização, que podem delegar a administração de sub-redes aos projetos. Os projetos podem ser explicitamente associados a sub-redes na VPC. Isso simplifica a configuração e torna a segurança e o controle de acesso mais transparentes.

TICK STACK é uma plataforma composta por componentes de código aberto, que facilita a coleta, armazenamento, geração de gráficos e alertas de dados de séries temporais, como métricas e eventos. Os componentes da TICK Stack são: Telegraf, um agente servidor para coletar e reportar métricas; InfluxDB, um banco de dados de séries temporais de alto desempenho; Chronograf, uma interface de usuário para a plataforma; e Kapacitor, um mecanismo de processamento de dados que pode processar, transmitir e agrupar dados do InfluxDB. Ao contrário de Prometheus, que é baseado no modelo Pull, TICK Stack se baseia no modelo Push de coleta de dados. O coração do sistema é o componente InfluxDB, que é um dos melhores bancos de dados de séries temporais. Essa stack é apoiada pelo InfluxData e, embora seja necessária a versão corporativa para recursos como clusterização de bancos de dados, ainda é uma boa opção para monitoramento. Estamos usando em alguns locais em produção e tivemos boas experiências.

O **AZURE DEVOPS** Services inclui um conjunto de serviços gerenciados, como repositórios Git hospedados, pipelines CI e CD e repositório de artefatos. O Azure DevOps Services substiuiu o Visual Studio Team Services. Tivemos uma boa experiência ao iniciar projetos rapidamente com os serviços do Azure DevOps – gerenciando, construindo e implantando aplicações para o Azure. Também nos deparamos com

alguns desafios – como a falta de suporte completo para pipelines de CI e CD como código, lentidão no tempo de inicialização de agente de compilação, separação de compilação e implantação em diferentes pipelines – e experimentamos algumas paralisações. Esperamos que o Azure DevOps Services melhore com o tempo para fornecer uma boa experiência de desenvolvimento ao hospedar aplicações no Azure, com uma experiência sem fricções na integração com outros serviços do Azure.

Inspirado no relatório do Google sobre Spanner – seu banco de dados distribuído baseado em relógios atômicos –, o CockroachDB é uma alternativa de código aberto que fornece transações distribuídas e particionamento geolocalizado, ainda suportando SQL.

(CockroachDB)

COCKROACHDB é um banco de dados distribuído de código aberto inspirado pelo relatório Spanner: o banco de dados distribuído do Google. No CockroachDB, os dados são automaticamente divididos em intervalos, normalmente de 64MB, e distribuídos entre nós no cluster. Cada intervalo tem um grupo de consenso e, pelo fato da plataforma usar o algoritmo de consenso Raft, os dados são sempre sincrcl 190.8sG seÞAR h.965 0.539 0.199 rg/aoTnDspirado nonseat2pDn rg8BtT4n.AÞJr1.2tcm Tc 7.54 0l70

Docker e fornece suporte experimental a Kubernetes. O gVisor é um projeto relativamente novo e recomendamos que ele seja avaliado para o panorama de segurança do seu contêiner.

Na maioria dos casos, blockchain não é o lugar certo para armazenar um arquivo blob (imagem ou áudio, por exemplo). Quando as pessoas desenvolvem aDApp, uma opção é colocar arquivos blob em algum armazenamento de dados centralizado, o que suporte experimen6gea91 0 Td lugi8mdo, o lvem

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 é uma ferramenta útil que ajuda pessoas desenvolvedoras a criar testes de ponta a ponta, e registra todas as etapas do teste como vídeo em um arquivo MP4 para facilitar a identificação de erros.

Uma ferramenta simples que impede que você faça commit de senhas e outras informações confidenciais em um repositório git, verificando também todas as revisões históricas antes de tornar um repositório público.

(git-secrets)

A segurança continua a ser primordial e a verificação negligente de credenciais e outros segredos no controle de códigos fonte é um importante vetor de ataque. **GIT-SECRETS** é uma ferramenta simples que impede que você faça commit de senhas e outras informações confidenciais a um repositório git. Ela também pode verificar todas as revisões históricas antes de tornar um repositório público, caso você queira garantir que nunca fez o check-in acidental de uma credencial. O git-secrets vem com suporte integrado para chaves e credenciais comuns da AWS e pode ser configurado rapidamente para outros provedores também.

HEADLESS FIREFOX tem o mesmo nível de maturidade do Headless Chrome para testes de front-end. Semelhante ao Headless Chrome, com o Firefox no modo headless, agora podemos aproveitar os testes do navegador sem os componentes visíveis da interface de usuário (UI), executando o conjunto de testes de UI com muito mais rapidez.
desempenhedorasr tes dea olocal

levar a plataforma IDEA completa ao .NET e aumentar a produtividade de quem está desenvolvendo. Independentemente da sua plataforma preferida, vale a pena explorar o Rider, uma vez que atualmente ele leva vantagem na produtividade em relação ao Visual Studio Code. Também é ótimo ver o ecossistema funcionando bem, já que a competição garante que essas ferramentas continuem melhorando.

SNYK te ajuda a encontrar, corrigir e monitorar vulnerabilidades conhecidas em árvores de dependência npm, Ruby, Python, Scala, Golang, .NET, PHP, Java e Docker. Quando adicionado ao seu pipeline de compilação, Snyk monitora e testa continuamente a árvore de dependências de bibliotecas em relação a um banco de dados de vulnerabilidades hospedado e sugere a atualização mínima de versão de dependência direta necessária para a correção.

À medida que mais times adotam DesignOps, as práticas e ferramentas nessa área amadurecem também. Muitos de nossos times agora trabalham com o que poderíamos chamar de **AMBIENTES DE DESENVOLVIMENTO DE UI**, que fornecem um ambiente abrangente para iterar rapidamente os componentes de interface de usuário, concentrando-se 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 forma autônoma no desenvolvimento de bibliotecas de componentes ou de sistemas de design, bem como incorporadas em um projeto de aplicação da Web. Em vez de criar o aplicativo, além de um BFF e serviços para adicionar um recurso a um componente, você pode iniciar o servidor dev do Storybook.

VISUAL STUDIO CODE é o editor/IDE gratuito da Microsoft, disponível para várias plataformas. Tivemos uma boa experiência ao usá-lo para desenvolvimento front-end com React, TypeScript e linguagens de back-end como GoLang, sem precisar alternar entre diferentes editores. O conjunto de ferramentas, o suporte a linguagens e as extensões do Visual Studio Code continuam crescendo e melhorando. Gostaríamos de destacar especificamente o VS Live Share para colaboração em tempo real e pareamento remoto. Embora projetos complexos em linguagens estaticamente tipadas, como Java, .NET ou C ++, provavelmente encontrem melhor suporte em IDEs mais maduros da Microsoft ou Jetbrains, acreditamos que o Visual Studio Code está se tornando cada vez mais uma ferramenta de escolha entre times de infraestrutura e desenvolvimento front-end.

A colaboração em tempo real com VS Live Share facilita o emparelhamento remoto. Particularmente, nós gostamos de que isso permita às pessoas desenvolvedoras colaborar usando seus próprios editores pré-configurados.
(VS Live Share)

VS LIVE SHARE é um conjunto de extensões para Visual Studio Code e Visual Studio. A colaboração em tempo real para edição e depuração de código, chamadas de voz, compartilhamento de terminal e exposição de portas locais reduziram alguns dos obstáculos que encontraríamos, caso contrário, ao parear remotamente. Em particular, gostamos do fato de o Live Share permitir que pessoas desenvolvedoras colaborem umas com as outras, enquanto continuam usando seu editor pré-configurado, que inclui temas, mapas de teclas e extensões.

Criar, testar e implantar aplicações móveis envolve várias etapas complexas, especialmente para um pipeline que vai dos repositórios de código-fonte até as lojas de aplicativos. Essa ferramenta fácil de configurar e específica de domínio pode reduzir a complexidade e a sobrecarga de manutenção.
(Bitrise)

Criar, testar e implantar aplicações móveis envolve várias etapas complexas, especialmente quando consideramos um pipeline que vai de repositórios de código-fonte até lojas de aplicativos. Todas essas etapas podem ser automatizadas com scripts e pipelines de compilação em ferramentas genéricas de CI/CD. No entanto, para times que se focam no desenvolvimento móvel e têm pouca ou nenhuma

CODEFRESH é um servidor de CI hospedado semelhante a CircleCI ou Buildkite. Ele é centrado em contêineres, tornando Dockerfiles e clusters de hospedagem de contêiner entidades de primeira classe. Gostamos do fato de a ferramenta incentivar uma abordagem de entrega com pipeline e oferecer suporte a branching e merging. Os primeiros relatórios de nossos times são positivos, mas ainda precisamos ver como isso funciona para projetos maiores e pipelines complexos.

Estamos sempre à procura de ferramentas e técnicas que permitam que os times de entrega trabalhem de forma independente do restante de uma organização maior, mantendo-se dentro de suas margens de segurança e risco. Grafeas é uma dessas ferramentas.

(Grafeas)

Estamos constantemente à procura de ferramentas e técnicas que permitam que os times de entrega trabalhem de forma independente do resto de uma organização maior, mantendo-se dentro de suas margens de segurança e risco. **GRAFEAS** é uma dessas ferramentas. Ela permite que as organizações publiquem metadados autoritativos sobre artefatos de software – imagens do Docker, bibliotecas, pacotes – que podem ser acessados a partir de scripts de compilação ou outros controles de conformidade automatizados. Os mecanismos de controle de acesso permitem uma separação de responsabilidades entre os times que publicam aprovações ou vulnerabilidades e os times que criam e implantam software. Embora várias organizações, incluindo Google e JFrog, usem Grafeas em seus fluxos de trabalho, vale notar que a ferramenta ainda está em versão alfa.

HEPTIO ARK é uma ferramenta para gerenciar recuperação de desastres para clusters Kubernetes e volumes persistentes. O Ark é fácil de usar e configurar e permite que você faça backup e restaure seus clusters por meio de uma série de pontos de verificação. Com o Ark, você pode reduzir significativamente o tempo de recuperação no caso de uma falha de infraestrutura, migrar facilmente os recursos do Kubernetes de um cluster para outro e replicar o ambiente de produção para testes e soluções de problemas. Ark suporta os principais provedores de armazenamento de backup (incluindo AWS, Azure e Google Cloud) e a partir da versão 0.6.0, um sistema de plugins que adiciona compatibilidade para plataformas adicionais de armazenamento de backup e volume. Os ambientes Kubernetes gerenciados, como GKE, fornecem esses serviços prontos para uso. No entanto, se você estiver operando Kubernetes localmente ou na nuvem, avalie com atenção o Heptio Ark para recuperação de desastres.

JAEGER é um sistema de rastreamento distribuído de código aberto. Semelhante ao Zipkin, foi inspirado pelo relatório Google Dapper e está em conformidade com OpenTracing. Jaeger é um projeto de código aberto mais novo que o Zipkin, mas ganhou popularidade rapidamente devido ao maior número de linguagens suportadas pelas bibliotecas clientes, bem como fácil instalação em Kubernetes. Usamos Jaeger com sucesso com o Istio, integrando rastreamentos de aplicações com o Envoy no Kubernetes e gostmos de sua interface do usuário. Com Jaeger se juntando o CNCF, prevemos um esforço maior de engajamento da comunidade e uma integração mais profunda com outros projetos CNCF.

KUBE-BENCH é um exemplo de analisador de configuração de infraestrutura que automatiza a verificação da sua configuração do Kubernetes em relação ao CIS benchmark para K8s. A ferramenta abrange autenticação de usuário, permissões e dados seguros, entre outras áreas. Nossos times têm considerado o kube-bench valioso na identificação de configurações vulneráveis.

Ark é uma ferramenta para gerenciar a recuperação de desastres para clusters Kubernetes e volumes persistentes. É fácil de usar e configurar, e permite fazer backup e restauração de seus clusters por meio de

(por exemplo, Kong), a comunidade .NET parece preferir o Ocelot para criar microsserviços. Em parte, o motivo é que o Ocelot se integra bem ao ecossistema .NET (por exemplo, com o IdentityServer). Outra razão pode ser que a comunidade .NET tenha estendido o Ocelot para suportar protocolos de comunicação como gRPC, Orleans e WebSocket.

Pesquisa de UX exigem coleta e análise de dados para melhores tomadas de decisões sobre os produtos que precisamos construir. O **OPTIMAL WORKSHOP** é um conjunto de ferramentas que ajuda a fazer isso de forma digital. Recursos como primeiro clique ou classificação de cartões ajudam a validar os protótipos e melhorar a navegação no site e a exibição de informações. Times distribuídos, em particular, beneficiam-se do Optimal Workshop, já que ele permite conduzir pesquisas remotas.

A extração de informações de negócios

Com a crescente adoção de [illegible]

MMKV é um framework de código aberto desenvolvido pelo WeChat, que fornece armazenamento rápido d

MockK é uma biblioteca nativa que ajuda nossos times a escrever testes limpos e concisos para aplicativos Kotlin, ao invés de usar empacotadores incômodos do Mockito ou do PowerMock..

(MockK)

MOCKK é uma biblioteca para mocking escrita em Kotlin. Suavemente

TYPESCRIPT é uma linguagem cuidadosamente considerada, e

de todas as bibliotecas ricas de JavaScript, ao mesmo tempo em que ganhamos segurança de tipagem. Isso é particularmente importante à medida que nossa base de código baseada no navegador continua a crescer. A tipagem no TypeScript permite usar IDEs e outras ferramentas para fornecer um contexto mais profundo ao seu código, além de fazer alterações e refatorar o código com segurança. O TypeScript, sendo um superconjunto de JavaScript, somado à documentação e à comunidade, ajudaram a facilitar a curva de aprendizado.

APACHE BEAM é um modelo de programação unificado de código aberto para definir e executar, em paralelo, pipelines em lote e streaming de dados. O Beam fornece uma camada de API portátil para descrever esses pipelines independentemente dos mecanismos de execução (ou runners), como Apache Spark, Apache Flink ou Google Cloud Dataflow. Diferentes runners têm diferentes capacidades e fornecer uma API portátil é uma tarefa difícil. Beam tenta encontrar um equilíbrio delicado, extraindo ativamente as inovações desses runners para o modelo Beam e também trabalhando com a comunidade para influenciar o roadmap desses runners. O Beam tem um rico conjunto de transformações de I/O integradas que cobre a maioria das necessidades de pipeline de dados. Ele também fornece um mecanismo para implementar transformações personalizadas para casos de uso específicos. A API portátil e as transformações de IO extensíveis formam um argumento convincente para avaliar o Apache Beam para necessidades de pipeline de dados.

Embora tenhamos uma tendência ao

Um insight que tivemos depois de conversar com nossos times é que Python está voltando para vários domínios tecnológicos. Na verdade, está a caminho de se tornar a linguagem de programação mais usada. Em parte, isso é impulsionado por sua adoção por cientistas de dados e em aprendizado de máquina, mas também vemos times adotando para construir microsserviços. **NAMEKO** é um framework de microsserviços super leve e uma alternativa a Flask para escrever serviços. Ao contrário do Flask, o Nameko possui apenas um conjunto limitado de recursos que inclui suporte para WebSocket, HTTP e AMQP. Também gostamos do foco na testabilidade. Se você não precisa de recursos como os templates que o Flask fornece, então vale a pena analisar Nameko.

POLLY.JS é uma ferramenta simples que ajuda os times a testar sites e aplicações em JavaScript. Nossos times gostam particularmente disso, permitindo que interceptem e façam stub de interações HTTP, o que permite um teste mais fácil e rápido do código JavaScript sem precisar aumentar os serviços ou componentes dependentes.

PREDICTIONIO é um servidor de aprendizagem de máquina de código aberto. Pessoas desenvolvedoras e cientistas de dados podem usá-lo para criar aplicatições inteligentes para previsão. Como todos as aplicações inteligentes, o PredictionIO tem três partes: coleta e armazenamento de dados, treinamento de modelos, implantação de modelos e serviço de exposição. As pessoas desenvolvedoras podem se concentrar na implementação de lógica de processamento de dados, algoritmo de modelo e lógica de previsão com base nas interfaces correspondentes, e liberar-se do armazenamento de dados e da implantação de treinamento de modelo. Em nossa experiência, o PredictionIO pode suportar volumes grandes e pequenos de dados com baixa conconrrência. Utilizamos o PredictionIO principalmente para criar serviços preditivos para pequenas e médias empresas ou como prova de conceito ao criar mecanismos de previsão mais complexos e personalizados.

No Radar anterior, mencionamos o Headless Chrome para testes de front-end. Com a adoção do Chrome DevTools Protocol (CDP) por outros navegadores, um novo conjunto de bibliotecas está surgindo para automação e testes em navegadores. O CDP permite um controle refinado sobre o navegador, mesmo no modo headless. Novas bibliotecas de alto nível estão sendo criadas usando o CDP para testes e automação. **PUPPETEER** é uma dessas novas bibliotecas. Ela pode conduzir o headless Chrome por meio de uma aplicação de página única, obter rastreamento de tempo para diagnósticos de desempenho e muito mais. Nossos times acharam mais rápido e mais flexível que as alternativas baseadas no WebDriver.

Enquanto ainda esperamos o hardware chegar, podemos experimentar a computação quântica usando linguagens e simuladores. As amostras do Q# podem

assíncrona. Ele oferece recursos de produtividade, como o recarregamento hot, e permite substituir partes da stack, por exemplo, a estrutura Web do lado do servidor ou o provedor de nuvem.

A adoção de uma nova linguagem normalmente provoca o surgimento de novas ferramentas que suportam práticas de engenharia maduras, como a automação de testes. Kotlin não é exceção. **SPEK** é um framework de testes – inspirada em ferramentas conhecidas, como Cucumber, RSpec and Jasmine – que cria testes em Gherkin e Specification, permitindo que os times tragam práticas maduras, como o desenvolvimento orientado por comportamento, para o espaço Kotlin.

Estamos testando **TROPOSPHERE** como forma de definir infraestrutura como código na AWS para nossos projetos nos quais AWS CloudFormation é usada no lugar de Terraform. troposphere é uma biblioteca Python que nos permite escrever código em Python para gerar descrições JSON em CloudFormation. O que mais gostamos em troposphere é que ela facilita a detecção rápida de erros JSON, aplicando verificação de tipo, testes de unidade e composição DRY de recursos da AWS.

WEBASSEMBLY é um grande avanço em termos de recursos de navegador como ambiente de execução de código. Suportado por todos os principais navegadores e compatível com versões anteriores, é um formato de compilação binária projetado para ser executado no navegador em velocidades quase nativas. Ele amplia a variedade de linguagens que você pode usar para escrever funcionalidades front-end, com foco inicial em C, C ++ e Rust, e também é um destino de compilação LLVM. Quando executado na sandbox, ele pode interagir com JavaScript e compartilhar as mesmas permissões e modelo de segurança. Quando usado com o novo compilador de streaming do Firefox, também resulta em inicialização de página mais rápida. Embora ainda seja cedo, este padrão W3C é definitivamente um padrão a se explorar.

Depois de trabalhar com um estilo funcional reativo em várias aplicações, nossos times ficaram impressionados e relataram que a abordagem melhora a legibilidade do código e o rendimento do sistema.

reasdigo

Saiba em primeira mão sobre o lançamento do próximo Technology Radar e atualize-se com webinars e conteúdos exclusivos.

INSCREVA-SE

thght.works/Sub-PT

ThoughtWorks®