Resenha do artigo Thoughtworks Technology Radar

Aluno: João Vitor Romero Sales
Disciplina: Projeto de Software
Professor: João Paulo Aramuni

Thoughtworks Technology Radar foco Adote

O artigo Technology Radar 30º volume, da empresa Thoughtworks se autodenomina “Um guia de opinião sobre o universo de tecnologia atual”. Este é dividido em duas partes, a primeira de Temas e a segunda do Radar. Na primeira parte são citados e discutidos temas abordados na última reunião do Conselho Consultivo de Tecnologia (em inglês, Technology Advisory Board - TAB), um grupo formado por 22 tecnologistas experientes da Thoughtworks. Já na segunda parte é criado um radar de Técnicas, Ferramentas, Plataformas e Linguagens e Frameworks, dividindo cada um destes em quatro estágios diferentes: Evite, Avalie, Experimente e Adote. Nesta resenha, apenas os blips que se encontram no estágio Adote serão considerados.

Um blip é uma tecnologia ou técnica que desempenha um papel no desenvolvimento de software. Os blips estão ‘‘em movimento’’, ou seja, não possuem posições fixas no radar. O artigo deixa claro que a confiança em recomendá-los cresce à medida que se movimentam entre os estágios.

Sobre os temas:

O ecossistema de código aberto, tradicionalmente baseado em licenças padronizadas pela Open Source Initiative (OSI), passa por um período de incerteza. A mudança abrupta de algumas ferramentas populares de licenças open source para modelos comerciais levanta preocupações sobre a sustentabilidade de projetos que se tornaram pilares da comunidade. Cobrar por funcionalidades extras é aceitável, mas a funcionalidade central, que sustenta todo um ecossistema, não deveria ser repentinamente bloqueada por um paywall. Essa prática, especialmente presente em ferramentas de IA, exige cautela por parte dos usuários. O artigo recomenda uma análise criteriosa das licenças, com atenção especial aos termos e condições para garantir que todos os componentes estejam sob a licença apropriada.

Nesse contexto de mudanças e incertezas, as IAs dominam as discussões sobre o futuro do desenvolvimento de software. Ferramentas como GitHub Copilot, CodiumAI e Continue prometem transformar a maneira como o código é escrito, revisado e testado. A integração da IA em todas as etapas do desenvolvimento impacta a dinâmica das equipes e a cultura de desenvolvimento, com potencial para aumentar a produtividade, a qualidade e a segurança, mas também trazendo novos riscos. A dependência excessiva da IA pode levar à negligência de boas práticas de engenharia, enquanto a falta de conscientização sobre seus perigos pode

comprometer a segurança do software. O artigo defende a adoção pragmática e responsável das IAs, explorando suas capacidades de forma crítica e equilibrando seus benefícios com os riscos.

A crescente popularidade dos LLMs (Modelos de Linguagem de Grande Porte) impulsiona o surgimento de padrões de arquitetura específicos para lidar com seus desafios e oportunidades. No artigo são destacadas ferramentas como o NeMo Guardrails para governança de LLMs, o Langfuse para aumentar a observabilidade e a Geração Aumentada por Recuperação (RAG) para aprimorar a qualidade das saídas. A definição de padrões e antipadrões é crucial para o amadurecimento e a disseminação do conhecimento sobre LLMs, e é prevista uma proliferação de novos padrões à medida que as IAs generativas se consolidam no desenvolvimento de software.

Em paralelo a essa ascensão das IAs, a Thoughtworks, defensora da Integração Contínua (CI), observa um movimento de afastamento dessa prática devido à crescente adoção de Pull Requests (PRs), originalmente concebidos para projetos de código aberto. Apesar de úteis para revisão de código, os PRs podem se tornar um gargalo no processo. O artigo explora soluções para aproximar os PRs dos princípios da CI, como ferramentas de gestão e automação (gitStream, Github Merge Queue), técnicas de granularidade e controle (diferenciais empilhados) e o uso de métricas para identificar gargalos. As IAs generativas também impactam a revisão de código, gerando PRs maiores e mais complexos, o que reforça a necessidade de buscar soluções que otimizem a revisão, garantindo a precisão da integração e a velocidade do feedback.

Sobre o radar:
Técnicas

Geração Aumentada por Recuperação (RAG) - É uma solução para aprimorar a qualidade das respostas geradas por Modelos de Linguagem de Grande Porte (LLMs).

Como funciona a RAG?

A RAG enriquece o contexto do LLM combinando a requisição do usuário com informações relevantes extraídas de um banco de dados externo. Este banco de dados armazena documentos confiáveis em diversos formatos (HTML, PDF etc.) e utiliza tecnologias como pgvector, Qdrant ou Elasticsearch Relevance Engine para garantir pesquisas eficientes.

Requisição: O usuário faz uma pergunta ao LLM.
Recuperação: O sistema pesquisa no banco de dados os documentos mais relevantes para a pergunta.

Combinação: Os trechos relevantes são combinados à requisição do usuário, formando um novo "prompt" enriquecido.
Geração: O LLM processa o "prompt" enriquecido e gera uma resposta mais completa e precisa.

Vantagens da RAG:

Respostas de alta qualidade: A RAG fornece ao LLM informações contextuais relevantes, resultando em respostas mais precisas e completas.
Redução de alucinações: LLMs podem gerar informações falsas ou sem sentido ("alucinações"). A RAG minimiza isto, fornecendo uma base sólida de informações reais.

Desafios da RAG:

Janela de contexto limitada: LLMs possuem um limite para o tamanho da entrada, logo selecionar documentos mais relevantes dentro desse limite é crucial.
Divisão de documentos: Documentos extensos precisam ser divididos em partes menores para o cálculo do "embedding". Garantir a sobreposição adequada entre as partes é um desafio importante.

Plataformas

CloudEvents - Eventos são amplamente utilizados em arquiteturas modernas, como as baseadas em eventos e serverless. No entanto, a falta de padronização entre provedores de nuvem e plataformas impede a interoperabilidade, dificultando a comunicação e integração entre sistemas.

Solução CloudEvents:

Interoperabilidade: Serviços, plataformas e sistemas podem compartilhar eventos de forma transparente, independentemente da linguagem de programação ou infraestrutura, uma vez que um formato comum para descrever dados de eventos foi definido.
Ampla Adoção: O CloudEvents é um projeto graduado da Cloud Native Computing Foundation (CNCF), indicando maturidade e forte apoio da comunidade.

Benefícios da Adoção do CloudEvents:

Portabilidade: Aplicações se tornam mais flexíveis e portáteis, podendo ser executadas em diferentes plataformas sem a necessidade de adaptar o tratamento de eventos.
Reutilização: Componentes e serviços podem ser facilmente integrados, aproveitando a padronização dos eventos.

Simplicidade: SDKs em diversas linguagens facilitam a adoção do padrão.

Aplicações do CloudEvents:

Plataformas de Nuvem Cruzadas: Integração simplificada entre diferentes provedores de nuvem.
Especificação de Eventos de Domínio: Criação de um vocabulário comum de eventos para domínios específicos.

Ferramentas

Conan - É uma ferramenta de código aberto para gerenciamento de dependências eficaz em projetos C/C++, e projetos destas linguagens frequentemente utilizam diversas bibliotecas de terceiros, o que pode tornar o gerenciamento de dependências complexo e propenso a erros. A falta de padronização e a variedade de plataformas e sistemas operacionais amplificam esses desafios.

Conan como Solução:

Facilidade: Definição, busca e gerenciamento de dependências de forma fácil e centralizada.
Suporte Multiplataforma: Compatível com os principais sistemas operacionais e diversas plataformas de destino (servidores, desktops, mobile, embarcados).
Criação e Publicação de Pacotes: Crie e publique suas próprias bibliotecas e pacotes C/C++.
Compartilhamento Eficiente: Compartilhe pacotes entre equipes utilizando servidores JFrog Artifactory.
Redução do Tempo de Compilação: Utilização de binários pré-construídos, especialmente útil para dependências complexas.
Integração com Ferramentas Populares: Integração com CMake e SDK Python para personalização do sistema de compilação.

Benefícios da Adoção do Conan:

Melhor Reprodutibilidade: Compilações consistentes em diferentes ambientes e ao longo do ciclo de desenvolvimento.
Ciclos de Desenvolvimento Mais Rápidos: Redução do tempo gasto com a resolução de problemas de dependências.
Bases de Código Mais Limpas: Código mais organizado e fácil de manter.

Kaniko - É uma ferramenta preferencial para a construção de imagens de container em pipelines baseados em container, impulsionada pela descontinuação do suporte ao Docker pelo Kubernetes em 2022, marcando uma mudança significativa no

cenário de desenvolvimento de aplicações em containers, demandando alternativas eficientes e seguras para a construção de imagens.

Kaniko como Solução:

Segurança: Kaniko constrói imagens de container sem a necessidade de um daemon Docker, eliminando vulnerabilidades relacionadas ao privilégio de root.
Eficiência e Performance: Projetado para ambientes Kubernetes, o Kaniko se integra perfeitamente a pipelines de CI/CD, otimizando o processo de construção de imagens.
Flexibilidade: Compatível com diversas ferramentas e configurações de pipelines, o Kaniko se adapta a diferentes fluxos de trabalho.

Adoção e Recomendação:

Satisfação com a Flexibilidade e Performance: Equipes reportaram experiências positivas com a ferramenta, elogiando sua capacidade de lidar com diferentes cenários e otimizar o processo de construção de imagens.
Recomendação como Padrão: A experiência positiva consolidou o Kaniko como ferramenta padrão para construção de imagens de container.

Karpenter - É um operador Kubernetes de código aberto para escalabilidade de nós, que oferece uma alternativa mais inteligente e flexível ao Cluster Autoscaler tradicional, que possui limitações em termos de flexibilidade e inteligência na escolha dos nós.

Este tipo de escalabilidade é crucial para aplicações Kubernetes, permitindo ajustar dinamicamente os recursos em resposta à demanda. No entanto, essa funcionalidade depende da disponibilidade de nós no cluster.

Sua crescente adoção entre provedores de nuvem e a capacidade de analisar workloads e otimizar recursos o tornam uma boa escolha para garantir a escalabilidade e o desempenho de aplicações em ambientes Kubernetes.

Karpenter como solução:

Análise Inteligente de Workloads: Examina as necessidades dos workloads e as restrições de agendamento para tomar decisões mais precisas sobre o provisionamento de nós.
Seleção Otimizada de Instâncias: Escolhe automaticamente o tipo de instância mais adequado às necessidades do workload, otimizando o uso de recursos e custos.
Provisionamento Dinâmico: Inicia e interrompe nós sob demanda, garantindo que a capacidade do cluster esteja sempre alinhada à demanda real.
Integração com Provedores de Nuvem: Embora inicialmente desenvolvido para AWS EKS, o Karpenter está se tornando um padrão entre os provedores de nuvem, com suporte recente ao AKS Karpenter Provider da Azure.

Vantagens do Karpenter:

Escalabilidade mais rápida e eficiente: A análise inteligente e a seleção otimizada de instâncias garantem que os pods sejam provisionados rapidamente e com os recursos adequados.
Redução de custos: O provisionamento dinâmico e a escolha inteligente de instâncias otimizam o uso de recursos e minimizam os custos.
Flexibilidade e Controle: Oferece mais opções de configuração e personalização em comparação ao Cluster Autoscaler tradicional.

Linguagens e Frameworks

Nenhuma recomendação