Briefing Técnico

A Survey of DevOps Concepts and Challenges

1. **Introdução**

DevOps é definido como um esforço organizacional colaborativo multidisciplinar com o objetivo de automatizar a entrega de atualizações, garantindo ao mesmo tempo a confiança da integridade da mudança. O movimento surge para implementar as lacunas de desenvolvimento e operações que funcionavam anteriormente com objetivos divergentes.

2. **Conhecimento**

A literatura de DevOps gira em torno de três pilares fundamentais:

- Colaboração
- Automação
- Medição e Compartilhamento

3. **Perspectiva**

- **Quebra de silos:** O objetivo é eliminar as barreiras entre os setores e alinhar as metas de toda a organização.
- **Mudança de Papéis:** A questão é se os desenvolvedores podem adquirir conhecimentos operacionais e se DevOps deve ser um cargo específico ou uma responsabilidade partilhada.
- **Estrutura**: A corporação deve decidir entre manter departamentos separados mas colaborativos, criar equipes multifuncionais ou estabelecer equipes de plataforma DevOps dedicadas.

4. **Problemas**

- **Exposição dos dados**: O artigo não explora a anonimização dos dados nas técnicas de automação. Ferramentas de logs criadas podem acidentalmente capturar dados pessoais identificáveis.
- **Cultura Hierárquica:** Muitas empresas mantém a cultura de punição e busca pelo culpado do erro, DevOps foca no conserto do erro sem a busca pelo culpado. Forçar a implementação da cultura DevOps sem a mudança da cultura da empresa fará com que erros apareçam mais rápidos, diminuindo o aprendizado.
- **Segregação de funções:** Embora o DevOps pregue que o desenvolvedor tenha controle da produção, artigos da LGPD muitas vezes exigem que quem desenvolve o código não tenha acesso aos dados sensíveis dos clientes.

5. **Ecossistema de Ferramentas e Automação**

As ferramentas não são o DevOps em si, mas são os meios que operacionalizam os seus conceitos:

- **Colaboração e Conhecimento**: Ferramentas como Slack e wikis de projeto ajudam a quebrar silos e facilitam a comunicação humana.
- **Pipeline de Entrega**: Ferramentas de CI/CD (ex: Jenkins, GitLab CI) e gestão de código (ex: Git) garantem a automação e a integração contínua.
- **Infraestrutura e Confiabilidade**: Tecnologias de contentorização (ex: Docker) e monitorização (ex: Prometheus) permitem que a infraestrutura seja tratada como código (IaC) e garantem a estabilidade em tempo de execução.

6. **Conclusão**

O artigo conclui que, embora a automação técnica esteja bem consolidada, o maior desafio reside na capacitação da colaboração humana e na definição da estrutura organizacional ideal. Recomenda-se que a gestão não foque apenas na compra de ferramentas, mas sim na criação de um ambiente onde a responsabilidade pela qualidade e estabilidade do software seja partilhada, enfrentando a quebra de silos e punição agregada a hierarquia cultural da corporação.