Metodologia CALMS

1- Diagnóstico Cultural

Descrição: O deploy é realizado manualmente no ambiente de produção, sem seguir um procedimento padronizado ou utilizar automação.

Problemas Identificados:

1. **Falta de Automação:** O deploy manual é propenso a erros humanos e inconsistências.
2. **Procedimento Não Padronizado:** Sem um processo padronizado, pode haver variação na forma como os deploys são realizados, resultando em falhas.
3. **Tempo Médio Elevado:** O tempo médio entre a entrega do código e o deploy é de 2 dias, o que indica um gargalo no processo.

Devido aos problemas descritos podemos identificar que atritos entre o setor de desenvolvimento e operações pode ser frequente. O setor de desenvolvimento não têm seus processos padronizados, já o setor de operações precisa de um procedimento diferente a cada entrega de valor/deploy. Estando sujeito a falhas de projeto e segurança, assim como retrabalho e atrasos.

Práticas de DevOps

1. Automação de Deploy (CD):

- **Ferramentas:** Utilizar ferramentas como Jenkins, GitLab CI/CD, CircleCI ou Azure DevOps para automatizar o processo de deploy.
- **Pipeline CI/CD:** Configurar pipelines de CI/CD que possam automatizar a construção, teste e implantação do código.
- **Benefícios:** Reduzir o tempo de deploy, diminuir a taxa de erros e aumentar a consistência.

2. Padronização do Processo:

- **Documentação:** Criar uma documentação detalhada e padronizada do processo de deploy.
- **Script de Deploy:** Desenvolver scripts de deploy reutilizáveis que garantam a consistência em cada implantação.

3. Testes Automatizados:

- **Integração de Testes:** Incorporar testes automatizados (unitários, de integração e end-to-end) no pipeline de CI/CD.

- **Benefícios:** Garantir que o código esteja funcionando conforme o esperado antes de ser implantado em produção.

4. Monitoramento e Feedback Contínuo:

- **Ferramentas de Monitoramento:** Implementar ferramentas de monitoramento contínuo como Prometheus, Grafana, ou New Relic.
- **Logs Centralizados:** Utilizar Elasticsearch para centralizar e analisar logs.
- **Alertas e Notificações:** Configurar alertas automáticos para incidentes e falhas, permitindo uma resposta rápida.

2- Automação

Automação de Deploy (CD):

- Através da implementação de uma esteira de deploy, usando um Github Actions por exemplo, permitindo ao time de desenvolvimento localizar e corrigir bugs, diminuindo atritos com o time de operações.

Padronização de Processos

- Através da padronização de processos, desde o desenvolvimento até os testes, é possível diminuir os atritos entres os times. Uma possível abordagem para se realizar isso é a elaboração de uma documentação ou instrução de trabalho com participação de ambos os times que vise de maneira equilibrada essa padronização.

3- Mensuração e Compartilhamento de Conhecimento

1. **Métricas de Eficiência**
    a. Tempo de de Ciclo
    b. Frequência de Deploys
    c. Tempo médio de de Implantação

2. **Métricas de Qualidade**
    a. Taxa de sucesso dos deploys
        i. Por porcentagem;
        ii. Deploys sem rollback ou fixes/hotfixes
    b. Número de Incidentes Pós Deploy
    c. Tempo médio de recuperação

3. **Métricas de Teste**
    a. Porcentagem de Cobertura de testes
        i. Normalmente setores de QA supervisionam cobertura de testes em aplicações, porém ainda sim é uma métrica a se avaliar
4. **Métricas de Monitoramento e Feedback**
    a. Tempo de Detecção de Incidentes
    b. Número de Alertas de Monitoramento
        i. Possível através de ferramentas de Telemetria como DataDog

5. **Ampliar o FeedBack**
    a. Integração de Feedback ao Ciclo de Desenvolvimento
        i. **Objetivo:** Centralizar e priorizar o feedback recebido.
        ii. **Formato:** Criar um backlog dedicado a melhorias contínuas baseado no feedback recebido.
        iii. **Processo:** Revisar e priorizar os itens do backlog durante as reuniões de planejamento de sprint.