DevOps

quinta-feira, 23 de dezembro de 2021 09:46

CAMS / CALMS

C = Cultura
A = Automação
M = Monitoramento
S = Compartilhamento (Sharing)

L = Lean

Três Maneiras

Pensamento de Sistemas

Amplificar laços de feedback

Cultura de Experimentação e aprendizado contínuos

Como nós precisamos encarar todos os processos, pessoas e todos os relacionamentos entres eles de maneira com que possamos aprimorar nossa maneira de desenvolver Software

No inicio era focado no time de Desenvolvimento, porem com o tempo percebeu-se que poderia agregar em outras áreas e setores

DevSecOps
=> Integrar melhor a area de desenvolvimento com Segurança
BizDevOps
=> Integrar melhor a area Bussiness com Desenvolvimento

Recado: FOQUE no CALMS

1ª Maneira :Pensamento de Sistemas

Sempre que penso em processo de otimização numa empresa eu não posso focar em uma area especifica e sim num todo.
Consequências:

- Buscar sempre aumentar o fluxo, mas cuidado pois isso pode acarretar em codigos complexos demais e lentos
- Nunca Passar defeitos para frente
- Nunca permitir que a otimização local crie degradação global, ou seja, equipe desenvolvimento entrega códigos com muita velocidade e sobrecarrega a equipe de QA, por exemplo.
- Alcançar uma compreensão profunda do sistema

2º Maneira: Laços de Feedback
Refletir sobre sua própria saída antes de ir ao próximo passo.

Compreender e atender melhor as necessidades do cliente
=> Capaz de Aprender oque está acontecendo em outras áreas e até com a experiência do usuário, na qual eu possa receber esse feedback o mais profundo e mais rápido possível

Garantir que as pessoas tenham as informações que precisam para atender melhor as necessidades

3ª Maneira: Experimentação e Aprendizado contínuos

Envolve a criação de uma cultura que promova a segurança para que o time experimente e arrisque com frequência e aprenda com essas experiências.

Precisamos ter a oportunidade de aprender o tempo todo.

Reserva tempo para melhoria de trabalho diário

Criar rituais que recompensem equipes por assumir riscos

Introduzir Falhas no sistema para aumentar a resiliência

LEAN
Modelo de gestão que busca remover os desperdícios enquanto aumenta a produtividade

7 Princípios:
Eliminar desperdícios

Inclua qualidade no processo => Refatoração codigo, code review, Pull request / Qualidade é responsabilidade do desenvolvedor

Crie Conhecimento => sessões de compartilhamento de conhecimento

Adie comprometimento (Decisões)

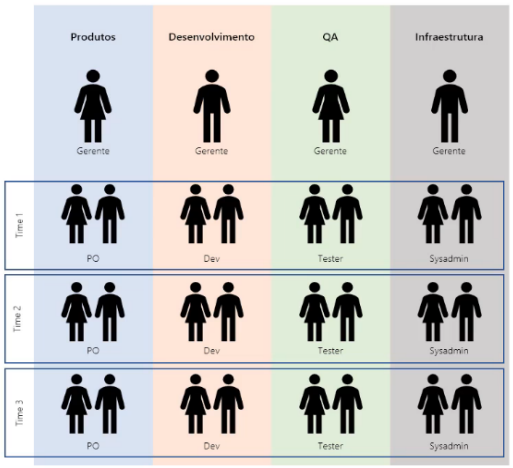
Entregue Rápido

Respeite as pessoas

Otimize o todo

Modelos de Times

- ✓Foco em times multifuncionais
  - ✓ Abordagem Ágil / Lean
  - ✓ Times requerem conhecimento diversificado e aprofundado
- ✓Prós
  - ✓ Reduz a necessidade de passagens de serviço
  - ✓ Facilita / acelera os laços de feedback
  - ✓ Desloca o controle de qualidade para "a esquerda"
- ✓Contras
  - ✓ Frequentemente incompatível com a estrutura de departamentos
  - ✓ Pode causar conflitos entre times e gestores
  - ✓ Requer buy-in de cima para mudanças profundas
  - ✓ "Accountability" menos claro na transição



Características de um time DevOps

Tipos de Trabalho :

- Divida técnica (Gambiarra)
- Trabalho Operacional (Estórias Técnicas)
- Trabalho não planejado (Imprevisto, interrupções)
- Radiadores de Informações (Kanbam)

Manter o Foco

- Definir Prioridade
- Liderança resolve os conflitos de prioridades
- Monitorar trabalho não-planejado
- BackLog unificado

Radiadores de Informações:

- Se possivel fazer a mão.

Cuidando moral do time:

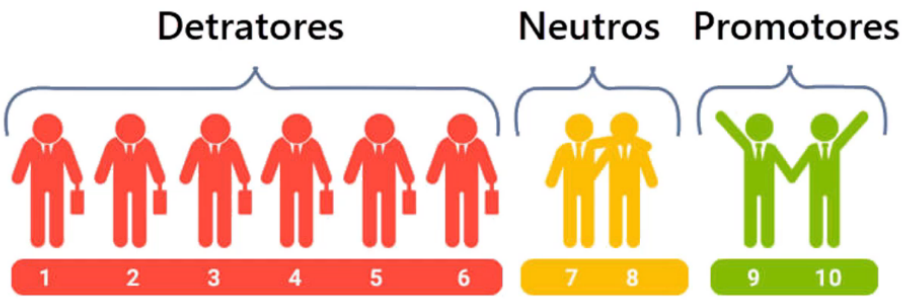
- Pessoas são nosso maior patrimônio
- O Papel do lider
  - Lider Servidor
  - Garantir Evolução do Time
  - Ajudar o time a encontrar sua motivação

Mantendo as pessoas motivdas:

- Por que
- Como
- O que

employee Net Promoter Score

- Medir a satisfação das pessoas do trabalho
- 2 perguntas
  - De 0 a 10 o quanto você recomendaria a sua empresa para um amigo ou familiar como um local para trabalhar?
  - De 0 a 10 o quanto você recomendaria o seu time para uma amigo ou familiar como local de trabalho?



Fórmula eNPS: %Promotores - %Detratores

100 Entrevistas:
- 27 Pessoas Detratoras
- 12 Pessoas Neutras
- 61 Pessoas Promotoras

61 - 27 = 34

100 Entrevistados
- 12 Detratoras
- 30 Neutras
- 58 Promotoras

58 - 12 = 46

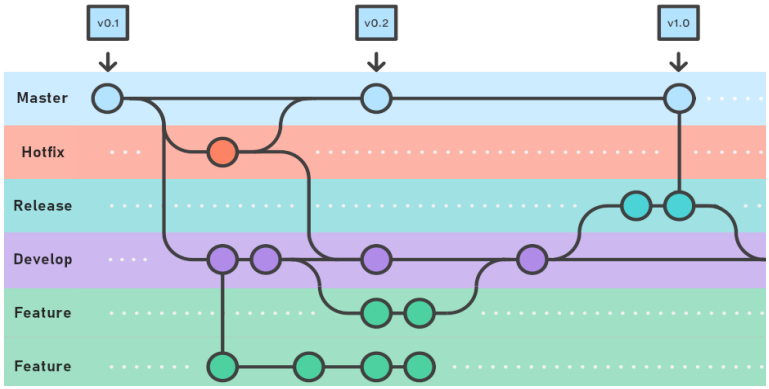
-100 -90 -80 -70 -60 -50 -40 -30 -20 -10 0 10 20 30 40 50 60 70 80 90 100

Altas Taxas de erros
Tendência a falhas
Pedidos de demissão

Times mais motivados e felizes
Time entende o seu valor
Alto desempenho e produtividade
Boas praticas
Compartilhe com outros times

Versionamento de código

Proteger Codigo-Fonte
- Manter Histórico
- Defazer alterações
- Desenvolvimento simultâneo
- Facil resolver erros
- Aditável
- Maior confiabilidade do produto

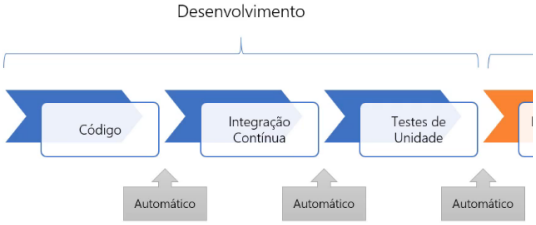


CI - Integração Contínua
- Integrar os codigos dos desenvolvedores com frequência

Implementar:
- Manter um repositório de código
- Automatizar Build
- Build Auto testáveis
- Todos os commits são feito na linha de base todos os dias
- Todo commit na linha de base é compilado numa máquina de integração
- Corrigir build quebrada imediatamente
- Manter os builds rápidos
- Tornar facil obter o executável mais recente
- Todo mundo pode ver o que está acontecendo
- Automatizar a implantação e testar num ambiente clone da produção

CD - Continuous Deployment
- Foca em garantir que o software está pronto para ser sempre que necessário
- Ênfase em testes automatizados
- Automatizar processo de implantação em diferentes a
- A liberação de versões para o usuário final é uma dec negócios após uma decisão manual

Continuous Delivery

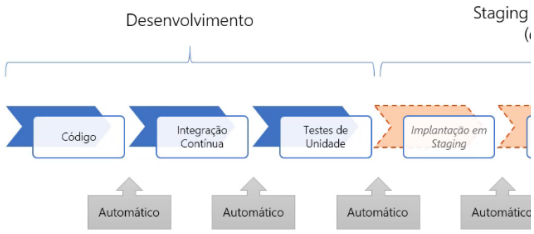


Replace
- Desliga ambiente atual => Substitui a versão e religa
- Simples de implantar
- Sempre atualizado
- Alto impacto no usuário (downtime)
- Rollback mais custoso

Roling Deployment

- Tira um servidor do LoadBalancer => faz implantação => quando concluído repete para todos os outros
- Sem downtime
- Fácil rollback
- Requer mais de um servidor
- Múltiplas versões simultaneamente
- Requer um bom roteamento de rede

Continuous Deployment



Blue/green Deployment

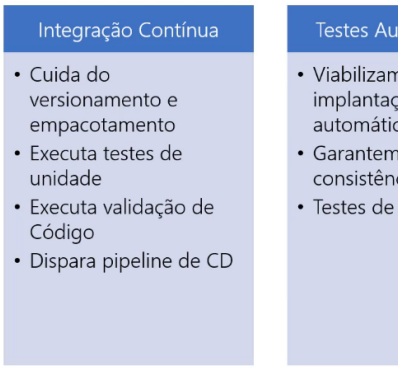
- Faz deployment no ambiente de staging => caso concluído, realiza um Swap
- Sem downtime
- Rollback instantâneo
- Mais caro, pois requer manutenção de dois ambientes

Rollback em caso de erros

Swap == Inverte o roteamento de rede, ou seja, os servidores d produção.

Canary Deployment

- Parecido com rolling deployment, porem sem remover o servidor da rede, poucos servidores é atualizado, e em caso de sucesso faz a implantação nos resto.
- Processo totalmente automatizado
- Sem dowtime
- Rollback simples
- Requer um processo maduro
- Depende de automação e testes
- Aplicação tem que ser capaz de funcionar com 2 versões diferentes ao mesmo tempo.



A/B testing deployment

- A nova parte é implantada em apenas uma parte dos servidores
- Permite 2 versões ao mesmo tempo
- Bom para validar hipóteses/experimentos
- Fácil rollback
- Requer processos maduro
- Aplicação tem que ser capaz de funcionar com 2 versões diferentes ao mesmo tempo.

Traffic Shadowing Deployment

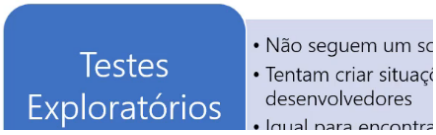
- Cria um ambiente de staging e redireciona uma "Cópia" de tráfego de rede para o novo ambiente.
- Permite testar requisições reais dos clientes
- Pode utilizar um Swap
- Sem dowtime
- Sem rollback
- Permiter testar com dados reais
- Ambiente complexo de montar e manter
- Mais caro, requer duplo ambiente de produção.

CI/CD não implica no uso de containers, por

Testes

- Entregar novas versões do produto o mais rápido possivel para o cliente
- Nem sempre automatizar um teste é viável.

- Qual o benefício de automatizar este teste?
- Devemos desenhar grandes testes fim-a-fim ou testes mais modulares?
- Qual a melhor maneira de automatizar este caso de teste?
- Como torno este teste mais robusto?
- O que mais podemos / devemos automatizar?



Monitoramento de Aplicações

- Monitoramento é essencial em DevOps
- Usado em conjunto com processos de testes, implantação e operação do ambiente de produção
- Pode ser feito no nível de infraestrutura ou de aplicação.

O que monitorar?

- Saúde do servidor
  - Uso Memória
  - Uso CPU
  - Espaço em disco
- Saúde da aplicação
  - Tempo de resposta
  - Taxa de erros
  - Quantidade de instancias
  - Disponibilidade
  - Requisições
- Segurança ambiente
  - Excesso de chamadas
  - comportamento suspeito
  - Login suspeito
- Atividade dos usuários
  - Sessões ativas
  - Fluxo de negócios
  - Falhas de login
  - Falhas de input
  - Dispositivos logados
  - Locais de acesso
- implantações
  - Ultima implantação
  - tempo de implantação
  - Tempo de rollback