Ontem e hoje na Qualidade de Software: Passagem de ambientes

Saiba sobre Workflow de desenvolvimento de Softwares: CI/CD e Pipelines

Salve jovem padawan, nossa conversa de hoje é um comparativo no fluxo de trabalho de um DEV, como sempre vou fazer uma analogia comparativa de como era em Mainframe e o Fluxo atual em DEVOPS.

O objetivo do artigo é trazer luz, falando sobre as boas práticas do mercado, desmistificando um processo tão delicado, em que DEVs novatos entram em pânico, como diria Douglas Adams não entrem em pânico!!!

Introdução

Primeiramente para terem uma base, irei contar como era no Banco Real, existiam 3 grupos de trabalho: Realplan (Analise e Desenvolvimento), RPD (Operações e produção) e um Grupo Misto Realplan e RPD (Performance).

Workflow de Desenvolvimento

Existia um aplicativo proprietário que garantia as versões de programa e a segmentação de Ambientes e gestionava o fluxo do fonte em cada ambiente, garantindo versões e possibilitando rollbacks e versões intermediarias em caso de necessidade.

O Sistema além disso guardava uma lista histórica de modificação (apenas comentários com 80 caracteres), 3 linhas de histórico da Guarda, utilizador RACF que copiou, alterou, guardou o programa.

Garantindo a segurança do código fonte assegurando uma trilha de auditoria para a DAI, inspecionar e analisar quaisquer desvios ou fraudes. Afinal o Sistema Bancario era auditado pela Febraban, Receita Federal, Bacen, CVM e outros órgãos do mercado financeiro.

Naquela época o sistema central era dividido em 3 Computadores: CA81 máquina de desenvolvimento e backup para Desastre Recovery em Campinas SP, maquinas de Produção, SP11 na Sede e SP 51 na IBM Tutoia.

Era um ambiente altamente burocrático onde existia padronização nos nomes de programas, nomes de variáveis, nomes de campos no Banco de Dados, nomes do CICS e telas 3270, composto por equipas multidisciplinares em comitê de telas, comitê de performance, comitê de relatórios, comitê de uso espaço em disco e memória.

O programa só avançava no workflow de passagem para a ambiente seguinte, quando todas as evidencias de testes fossem anexadas no diretório de documentação, existiam inúmeras rotinas comuns no Ambiente, códigos com manuais de desenvolvimento com copybooks padronizados.

Nenhum programa avançava para Produção sem o aval dos comitês e em alguns casos, voltavam para o Ambiente de Desenvolvimento comum memorando duro avisando da anomalia detectada.

A vantagem disso era que os sistemas eram parrudos com poucos abends e uso controlado de recursos (impressora, discos e fitas, memoria) dando confiabilidade ao desenvolvedor e segurança ao usuário, por saber que o software iria apresentar resultados fidedignos e seguros, segmentos em ambientes, estanques e isolados.

Ambiente de Desenvolvimento

Área livre para desenvolvedor com acesso a rotinas públicas e base de dados sem consistência de negócio destinada a testes unitários.

Ambiente de Aceite

Área controlada com processos batch para sincronizar dados e simular o processo de produção.

Ambiente de Produção

Arear estrita e sem back-door de desenvolvimento com rotinas batchs diárias, semanais, quinzenais, mensais, bimestrais, semestrais e anuais controladas pelo Control-M, uma poderosa ferramenta de scheduling.

Todos os ambientes protegidos pelo RACF, com rotinas de log de operações CRUD, logs na base de dados Adabas e Db2, controle de acesso aos arquivos VSAM e GDG sequenciais, com navegação nas telas obedecendo as normas de auditoria e sistema de autorização com dual users. Usando o software Predict da Software AG como repositório e dicionário de dados facilitando consulta e pesquisas, trabalhando ativamente em referência cruzada de programas copybooks/includes e tabelas de base de dados.

Esse era o sistema informático que funcionou no Banco Real de sua fundação até a sua extinção, quando foi incorporado pelo ABN Amro Bank em 1997 e paulatinamente sumiu do cenário com a aquisição pelo Banco Santander em 2007 e completa eliminação em 2010.

Fora das grandes instituições o caos reinava

Sabemos que no mundo exterior, fora das estruturas hierarquizadas e burocráticas dos grandes centros de processamento de dados civis e militares, dos grandes bancos e agências governamentais a história era outra.

Gastaram milhares de horas e milhões de páginas impressas tentando normatizar e as melhores cabeças criaram inúmeras metodologias e técnicas para melhorar o trabalho de desenvolvimento de software na vã tentativa de garantirem a qualidade.

A seguir falaremos da bala de prata, que auxilia na maioria dos problemas no desenvolvimento de software e sua respectiva qualidade final ao usuário, que apenas deseja apertar o botão e ter a solução na palma da mão, ou melhor, tela do celular, mas o troll backlog sobrevive.

BACKLOG

A famigerada lista sem fim de pedidos de funcionalidades e correções de pequenas anomalias, infelizmente por melhor que seja a metodologia, infelizmente o usuário seria mais rápido solicitando melhorias, correções e evoluções de softwares.

Respeitável Padawan apresento CI , CD, Pipelines

Em informática chegara um dia, que faltaram siglas, acrônimos e outras meios para identificar tecnologias, metodologias e afins, por hora vamos conhecer melhor este novo conceito, resumidamente com o advento das metodologias ágeis com “velocidade na entrega” e melhor qualidade final do software.

Isto é menos programadores, menores custos no uso de maquinas e equipamentos, melhor documentação e melhor qualidade do software adequado aos padrões e melhores praticas.

Entregas continuas

É uma pratica de entrega de programas, utilizando técnicas agiles de modo que cada Sprint gere entregáveis funcionais, utilizando o conceito do MVP ([Minimum Viable Productl](https://www.google.com/search?sxsrf=AOaemvKnwE8pj_P1-1MaJb92jtWXYIwQ3Q:1642733296047&q=minimum+viable+entreg%C3%A1vel&spell=1&sa=X&ved=2ahUKEwjeoMi06sH1AhWtqpUCHcOoDlwQBSgAegQIARA3)), com isso o usuário está sempre contente e tem novidades para explorar e descobrir as funcionalidades.

Uma alteração feita por um desenvolvedor é enviada para um repositório, onde é consolidada com outras alterações e o package gerado evolui para outros ambientes seguindo o workflow, ou melhor dizendo entra em uma pipeline.

O que é pipeline?

Padawan são tantos conceitos e as vezes devo desviar ligeiramente do objetivo para esclarecer alguns temas, na era mainframe era usual dizer Estagio, Stage num workflow, ou fluxo de trabalho. Com a evolução dos termos, surgimento de novas tecnologias e roupagem cool, surgiu o termo pipeline, tomado de empréstimo a indústria petrolífera, isso mesmo, em bom português poderíamos dizer oleoduto.

Simplificando o artigo, pipeline é o conduto que organiza os processos CI/CD, de passagem de programas, com algumas ferramentas adicionais como controle de versão, merge de fontes, auditoria e controle de qualidade tudo em uma única metodologia. Existem duas maneiras a manual e a automatizada, mais a frente apresento uma ferramenta bem famosinha.

O que é CI?

Sigla para Integração continua, pense que assim que novas alterações no código são efetuadas, compiladas, testadas e refinadas, sendo enviadas para um repositório estilo Github e compartilhado com o restante da equipe. Uma solução necessária para evitar conflitos de versões, conflitos nas ramificações quando muitos programadores e equipes diferentes manipulam o código.

O que é CD?

Uma sigla coringa que significa dois estágios na linha de produção dos softwares, pode referir-se à implantação continua e também a entrega continua, calma não me confundi e que na verdade ambas as formas descrevem maneiras de automatizar as diversas fases na evolução do código fonte durante as fases seguintes do workflow do pipeline.

As duas situações previstas no CD servem para automatizar o processo de deploy do software, tornando a guarda de programas e passagem entre ambientes sejam mais rápidas e confiáveis, minimizando os riscos de erro humano e facilitando o merge de trabalho para equipes grandes.

O risco de erro humano é grande, em trabalhos com centenas de códigos fontes com elevado grau de referência cruzada, os pacotes de passagem necessitam de automação, em resposta a essa dor. Surgiu um software open-source para gerir estes pacotes conheça o Jenkins.

O que é Jenkins?

Uma das ferramentas mais populares para a automação dos processos de criação de software, automatizando tarefas chatas e repetivas, atividade que a atenção muita esvai-se e local onde ocorrem erros fatais, que colocam em xeque o trabalho de uma equipe inteira.

O Jenkins foi criado por Kohsuke Kawaguchi, uma solução encontrada por ele, a fim de evitar erros no trabalho, diz a lenda, que Kawaguchi está cansado de ser acusado por erros na execução das atividades, e nas suas horas vagas codificou o Hudson no princípio da década de 2000.

A ideia deu tão certo, que o Hudson tornou-se uma ferramenta comercial, para desgosto de Kawaguchi, que desejava uma ferramenta democrática, livre e acessível para todos os DEVS do planeta, por isso numa atitude ousada liberou o código fonte, através da comunidade open-source surgia o Jenkins, que desde 2004 não para de receber novas atualizações e plug-ins.

Afinal o que o Jenkins faz?

Imagine que toda a orquestração da pipeline, bem como documentação, passagem de fonte e entrega de software para todos os estágios intermediários, finalizando com o Deploy e acompanhamento do pós-produção, mas isso é assunto para outro artigo, convido a conhecerem o Jira.

Conclusão

Querido padawan divaguei bastante sobre o tema o que acontece após codificar, testar e entregar o software. Nesta etapa do trabalho nos tempos atuais tratamos sobre o CI/CD e sua pipeline com ajudinha do Jenkins.

Ao mesmo tempo fiz um flashback de como era desenvolvida a mesma atividade na época dos mainframes e uma pequena homenagem a meus amigos do Banco Real, pessoas fabulosas que partilhavam seu conhecimento com os colegas, eram os parças, obrigado Tokunaga, Honda, Mugnol, Bob, Marcio Silva, Alice, Roseli, Darci, Kuni e tantos outros. Passados quase 3 décadas guardo com carinho a amizade e companheirismo.