CI

Continuous Integration

Aula 1

Professor: Tiago Lage Payne de Pádua

❑ Bibliografia

ATLASSIAN

https://www.atlassian.com/continuous-delivery/continuous-integration

https://martinfowler.com/articles/continuousIntegration.html

"A Integração Contínua (CI) é o processo de automatizar a criação e o teste de código toda vez que um membro da equipe entrega alterações no controle de versão. O CI incentiva os desenvolvedores a compartilharem seus códigos e testes de unidade mesclando suas alterações em um repositório de controle de versão compartilhado após cada pequena conclusão de tarefa. O código de confirmação aciona um sistema de compilação automatizado para capturar o código mais recente do repositório compartilhado e para construir, testar e validar a *branch* principal."

❑ Testes automatizados (Automated tests)

- Testes automatizados são testes que podem ser executados sem a necessidade de intervenção humana de maneira repetitiva, a qualquer momento;
- Normalmente tem-se que escrever um script para testar algumas afirmações ou validar o comportamento do seu aplicativo;
- O script é então executado por uma máquina que fornece os resultados como uma saída;
- O teste automatizado é uma parte essencial do CI, mas não é suficiente por si só.

❑ Entrega contínua (Continuous delivery)

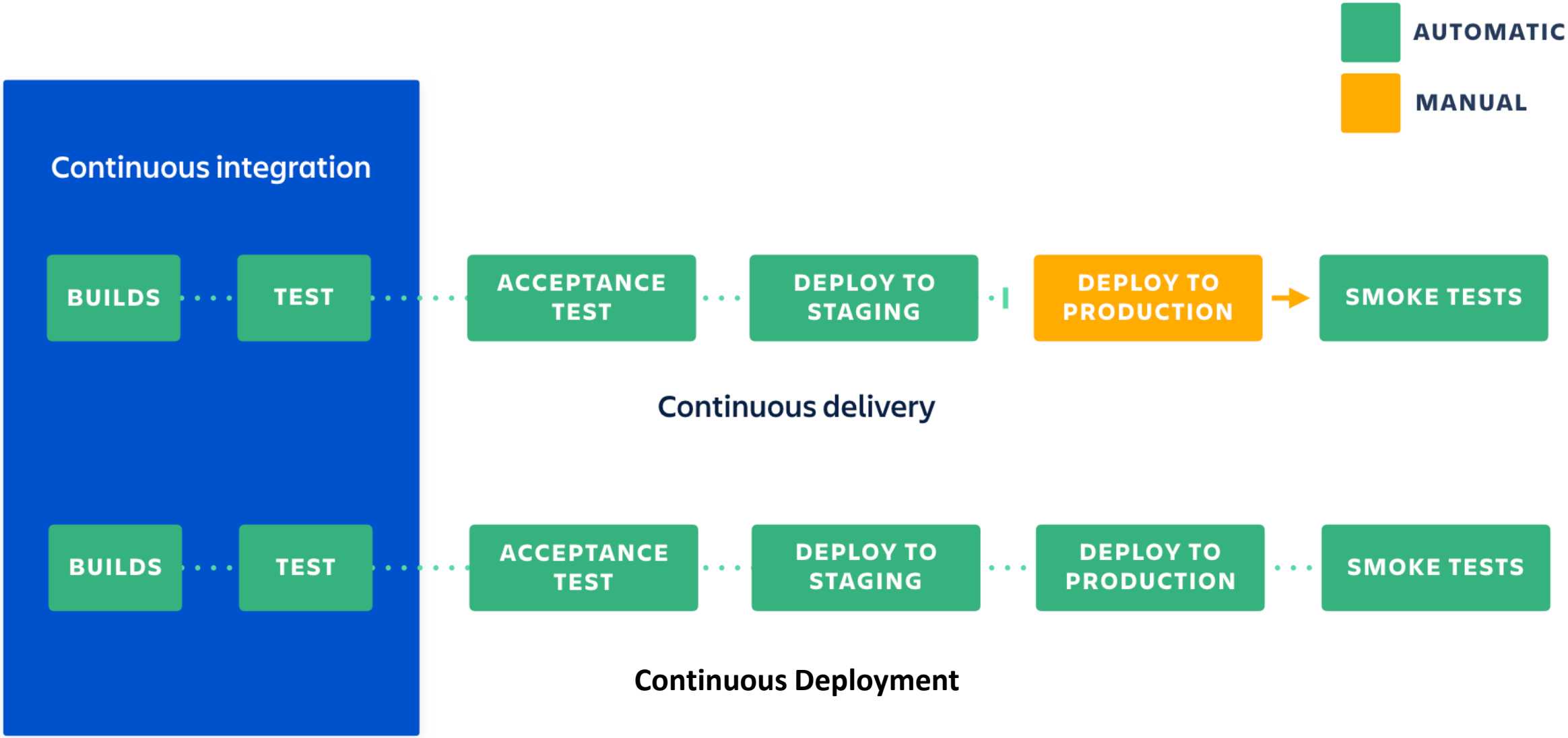
- É uma extensão da integração contínua para garantir que se possa liberar novas alterações para os clientes rapidamente e de maneira sustentável;

- Isso significa que, além de automatizar os testes, também automatiza-se o processo de liberação e pode-se implantar o aplicativo a qualquer momento;

- Em teoria, com a entrega contínua, pode-se decidir entregar diariamente, semanalmente, quinzenalmente, ou o que for adequado às necessidades comerciais. No entanto, se realmente deseja-se obter os benefícios da entrega contínua, implanta-se na produção o mais cedo possível para ter certeza de liberar pequenos lotes que sejam fáceis de solucionar em caso de um problema;

❑ Implantação Contínua (Continuous deployment)

- A implantação contínua vai um passo além da entrega contínua;
- Com essa prática, todas as alterações que passam por todos os estágios do pipeline de produção são liberadas para os clientes;
- Não há intervenção humana e somente um teste com falha impedirá que uma nova mudança seja implantada em produção;
- A implantação contínua é uma excelente maneira de acelerar o ciclo de feedback com os clientes e tirar a pressão da equipe, já que não há mais um dia de implantação. Os desenvolvedores podem se concentrar na criação de software, e eles veem o seu trabalho implantado minutos depois de terem terminado de trabalhar nele;



Continuous Deployment

❑ Requisitos da integração contínua

- A equipe precisará escrever testes automatizados para cada novo recurso, melhoria ou correção de bug;
- É necessário um servidor de integração contínua que possa monitorar o repositório principal e executar os testes automaticamente para cada novo *commit* enviado;
- Os desenvolvedores precisam mesclar suas alterações com a maior frequência possível, pelo menos uma vez por dia;

❑ Benefícios da integração contínua

- Menos bugs são enviados para produção à medida que os erros são capturadas antecipadamente pelos testes automatizados;
- Construir o a versão de implantação é fácil, pois todos os problemas de integração foram resolvidos cedo;
- Menos "troca de contexto", pois os desenvolvedores são alertados assim que quebram a compilação e podem trabalhar para corrigi-la antes de passar para outra tarefa;
- Os custos de teste são reduzidos drasticamente – o servidor de CI pode executar centenas de testes em questão de segundos;
- A equipe de controle de qualidade passa menos tempo testando e pode se concentrar em melhorias significativas na cultura de qualidade;

❑ Requisitos da entrega contínua

- É necessário uma base sólida em integração contínua e sua suíte de testes precisa cobrir o suficiente da sua base de código;
- Implantações precisam ser automatizadas. O acionador ainda é manual, mas uma vez iniciada a implantação, não deve haver necessidade de intervenção humana;
- A equipe provavelmente precisará incluir sinalizadores de funcionalidades para que as funcionalidades incompletas não afetem os clientes na produção;

❑ Benefícios da entrega contínua

- A complexidade da implantação de software é removida. O time não precisa mais passar dias se preparando para uma implantar;
- Pode-se implantar com mais frequência, acelerando assim o ciclo de feedback com seus clientes;
- Há muito menos pressão nas decisões para pequenas mudanças, incentivando, portanto, a interação mais rápida;

❑ Requisitos da implantação contínua

- A cultura de teste precisa estar no seu melhor. A qualidade do conjunto de testes determinará a qualidade dos seus lançamentos;
- O processo de documentação precisará acompanhar o ritmo das implantações;
- A marcação de funcionalidades se torna parte inerente do processo de liberação de mudanças significativas para garantir que se possa coordenar com outros departamentos (Suporte, Marketing, RP...);

❑ Benefícios da implantação contínua

- Pode-se desenvolver mais rápido, pois não há necessidade de pausar o desenvolvimento para implantações. Os pipelines de implantação são acionados automaticamente para cada alteração;

- As liberações são menos arriscadas e mais fáceis de corrigir em caso de problema, à medida que se implanta pequenos lotes de alterações;

- Os clientes veem um fluxo contínuo de melhorias e a qualidade aumenta a cada dia, em vez de todo mês, trimestre ou ano;

❑ Proteção da qualidade com construções contínuas e automação de testes

- **Construções contínuas:** construir o projeto assim que uma alteração é feita. Idealmente, o delta entre cada compilação é um único conjunto de mudanças;

- **Automação de teste:** validação programática do software para garantir a qualidade. Os testes podem iniciar ações no software a partir da interface do usuário ou da camada de serviços de back-end.

- A integração contínua combina construções contínuas com automação de testes para garantir que cada construção também avalie a qualidade da base de código.

❑ Testes no CI: testes unitários, API e funcionais

- As execuções de IC têm duas fases principais;

1. O primeiro passo garante que o código seja compilado. (Ou, no caso de linguagens interpretadas, simplesmente une todas as partes.)

2. A segunda etapa garante que o código funcione conforme projetado. A maneira mais certa de fazer isso é com uma série de testes automatizados que validam todos os níveis do produto;

❑ Testes unitários

- **Benefícios:** Fácil de escrever, executar rapidamente, modelar de perto a arquitetura da base de código;

- **Desvantagens:** os testes unitários apenas validam os principais componentes do software; eles não refletem fluxos de trabalho do usuário que geralmente envolvem vários componentes trabalhando juntos;

❑ Testes de API

- Um bom software é modular, o que permite uma separação mais clara do trabalho em várias aplicações. As APIs são os pontos finais nos quais diferentes módulos se comunicam entre si, e os testes da API os validam fazendo chamadas de um módulo para outro.

- **Benefícios:** Geralmente fácil de escrever, executar rapidamente e pode facilmente modelar como os aplicativos irão interagir uns com os outros.

- **Desvantagens:** em códigos simples, os testes de API podem imitar alguns testes de unidade;

❑ Testes Funcionais

- Os testes funcionais funcionam em áreas maiores da base de código e modelam fluxos de trabalho de usuários. Em aplicativos da Web, por exemplo, HTTPUnit e Selenium interagem diretamente com a interface do usuário para testar o produto.

- **Benefícios:** mais propensos a encontrar erros, pois eles imitam as ações do usuário e testam a interoperabilidade de vários componentes.

- **Desvantagens:** mais lentas que os testes de unidade e, às vezes, relatam falsos negativos devido à latência da rede ou a uma interrupção momentânea em algum ponto da pilha tecnológica;

Ano: 2015 Banca: CESPE Órgão: TCU Prova: CESPE - 2015 - TCU - Auditor Federal de Controle Externo - Tecnologia da Informação

Acerca de integração contínua e entrega contínua, julgue o próximo item.

Para que a prática de integração contínua seja eficiente, é necessário parametrizar e automatizar várias atividades relativas à gerência da configuração, não somente do código-fonte produzido, mas também de bibliotecas e componentes externos.

( ) Certo
( ) Errado

Ano: 2015 Banca: CESPE Órgão: TCU Prova: CESPE - 2015 - TCU - Auditor Federal de Controle Externo - Tecnologia da Informação

Acerca de integração contínua e entrega contínua, julgue o próximo item.

Para que a prática de integração contínua seja eficiente, é necessário parametrizar e automatizar várias atividades relativas à gerência da configuração, não somente do código-fonte produzido, mas também de bibliotecas e componentes externos.

(x) Certo

( ) Errado