1. Gerenciamento do processo de construção/build (mapeamento para ferramentas para ambientes de desenvolvimento e produção)

O gerenciamento de construção e liberação é o processo de gerenciamento, planejamento, programação e controle de uma construção de software em todo o seu ciclo de vida.

A construção de um aplicativo ou software envolve vários estágios. Cada build possui diferentes números de build e é sempre construído a partir de um repositório de código-fonte como o git. Construir um aplicativo ou software requer ferramentas de construção como ant, maven, gradle, etc. As ferramentas de construção compilam os arquivos de código-fonte em arquivos ou pacotes executáveis reutilizáveis.

O gerenciamento de construção e liberação controlará o ciclo de vida de um produto de software, o processo de planejamento, gerenciamento, programação e controle da construção em diferentes estágios e ambientes, como desenvolvimento, teste, preparação e produção

Existem 5 tipos de ambientes na infraestrutura de construção e lançamento:

- **Dev:** Equipe de desenvolvimento mantém este ambiente para escrever seu código. Apenas a equipe de desenvolvimento tem acesso a este ambiente. QA ou outras equipes não têm acesso a este ambiente. A equipe de desenvolvimento usa esse ambiente, principalmente para escrever casos de teste de unidade.
- **QA: o** ambiente de QA é propriedade da equipe de teste e o teste real ocorre neste ambiente. A equipe DEV não tem acesso ao ambiente de controle de qualidade. Após a conclusão da codificação, o código é movido do ambiente DEV para o ambiente de controle de qualidade para a execução do teste.
- **UAT:** O ambiente de teste de aceitação do usuário é usado por usuários de negócios para realizar testes após a conclusão dos testes do sistema. Nesta fase, o produto é testado do ponto de vista do negócio. O acesso ao ambiente UAT é restrito apenas aos usuários de negócios e, em algumas ocasiões, o acesso temporário será fornecido à equipe de QA se, no caso de usuários de negócios, precisarem de assistência.

- **Staging:** O ambiente STAGING se parece exatamente com o ambiente de produção. O aplicativo instalado no ambiente de teste deve corresponder exatamente ao ambiente de produção.
- **Prod:** O ambiente PROD é o ambiente de produção que é acessado por usuários reais e nenhuma das equipes DEV e QA tem acesso a este ambiente.

2. Entrega contínua

Entrega contínua é uma prática de desenvolvimento de software que utiliza a automação para acelerar o lançamento de novo código.

Ela estabelece um processo pelo qual as alterações feitas por um desenvolvedor em uma aplicação possam ser enviadas para um repositório de código ou um registro de aplicações em container por meio da automação.

A entrega contínua é parte da CI/CD, um método de entrega de software usado com frequência ao automatizar algumas das fases de desenvolvimento de aplicações.

"CI" refere-se à integração contínua. Com a integração contínua, novas alterações no código de uma aplicação são criadas, testadas e mescladas em um repositório compartilhado. É a solução ideal para evitar conflitos entre ramificações quando muitas aplicações são desenvolvidas ao mesmo tempo.

“CD” pode se referir à implantação contínua ou à entrega contínua, que descrevem formas de automatizar as fases posteriores do pipeline.

3. Ferramenta de Apoio à Integração Contínua

As 7 principais ferramentas de integração contínua que serão citadas abaixo, vão conseguir atender a maioria das necessidades.

Jenkins: Trata-se talvez da ferramenta principal de código aberto para integração continua, pois é a mais famosa e utilizada. A mesma ocupa a primeira posição no DevOps das ferramentas, obviamente que na seção de integração.

A grande vantagem é que o código é aberto e ao mesmo modular (a sua funcionalidade de base line é um pouco limitada, existindo mais de 900 plug-ins que vão estender os recursos fornecendo integrações adicionais.

É uma ferramenta que é totalmente baseada em Javas, portanto, contempla todas as suas extensões. Sem esquecer dos plug-ins que serão desenvolvidos em Java, portanto, trata-se de uma ferramenta que é um pouco colaborativa.

Já que a plataforma é a famosa Java RuTime, o Jekins consegue ser instalado em qualquer tipo de sistema operacional. Desde que o mesmo já tenha o Java executando, ou seja, funcionando perfeitamente e sem dar nenhum problema.

Em regra geral, essa é uma das principais ferramentas de integração contínua pois é feito em cima de uma comunidade. É simples de modificar os plug-ins é muito mais fácil, portanto, consegue ser algo bem vantajoso por isso. Todos sabem que faz muita diferença encontrar suporte facilmente na internet.

Travis: Essa ferramenta é de código de aberto e pode ser utilizada em todos os projetos que forem hospedados no GitHub. Só que é preciso citar, que uma vez hospedado o mesmo não irá depender de qualquer outra plataforma. É primordial guardar essa informação, porque no final fará diferença de uma forma positiva.

O mesmo precisa ser configurado de uma forma especifica e precisará utilizar os arquivos travis.yml. Esses arquivos vão precisar conter as informações, porque o Travis consegue suportar uma variedade grande de diferentes linguagens.

Essa configuração da compilação vai precisar ser bem documentada, ou seja, é preciso ter um editor de texto instalado. É necessário citar que o Travis utiliza as máquinas virtuais para conseguir construir as aplicações.

É solução mais interessante para pessoas que tenham uma repositória privado ou mesmo precisem executar mais de uma tarefa ao mesmo tempo. Vão existir vários planos referentes as assinaturas mensais, ou seja, escolha a melhor. Será necessário cumprir alguns requisitos veja a seguir quais são:

- Ter instalado o Bower, NodeJS e Gir;

- E possível usar a mesma conta no GitHub para acessar o Travis;
- Possuir uma conta no GitHub e Heroku

GitlabCI: Uma das principais ferramentas de integração contínua é essa, porque a eficiência é grande. O mesmo provê suporte para a serie continuous (integration, deployment e delivery), só que o foco é maior no primeiro. Os demais são importantes, porém o que fará diferença é justamente o primeiro.

O objetivo principal dessa ferramenta é que em relação ao commit é necessário que o mesmo seja integrado ao repositório central. O desenvolvedor precisa garantir que o mesmo não quebre os testes automatizados e nem a compilação.

É preciso mostrar que para cada commid, o GitlabCI vai rodar um pipeline de integração, passando por estágios. Cada um será diferente de acordo com o projeto, ou seja, irá compilar e fazer a execução de testes automatizados.

Logo após o mesmo ser compilado e testado com todo o sucesso, é preciso fazer a construção de uma imagem Docker. Sem esquecer também de fazer o deploy em um ambiente que seja de produção ou mesmo aceite.

O cuidado maior é na hora de fazer a implementação, pois é preciso realizar alguns commits que sejam coerentes. Os desenvolvedores saberão como fazer, pois, a ferramenta é autoexplicativa (ao menos é o que dizem).

Hudson: Falar sobre o Hudson sem citar o Jenkins é praticamente impossível, porque ambos estão diretamente ligados. O projeto do Jenkins originou-se de uma divisão do Hudson, logo após de uma disputa com a Oracle. Não é à toa que ambos são bem parecidos, indo desde a aparência até as funcionalidades.

Sem dúvidas, uma das principais vantagens do Hudson é que o mesmo permite realizar testes durante a integração continua. Uma vez que os testes unitários podem exigir muito tempo e são complexos em demasia, o que é ruim.

O Hudson é conhecido como um servidor de testes, ou seja, é possível receber vários projetos. Não importando o servidor internos ou mesmo externos, que vão fazer build de todos esses projetos que estiverem sendo realizados.

É possível construir e também integrar vários tipos de projetos sem que existam maiores problemas. A principal vantagem é que vários desenvolvedores podem fazer a utilização do mesmo servidor e também do Hudson.

Além disso, é permitido trabalhar em muitas tarefas que são distintas, sem contar os diversos recursos automatizados. Por exemplo: atualizando a sua cópia de trabalho antes de começar a execução da sua compilação. Evitando que um projeto desatualizado seja compilado na mesma versão e complique tudo.

É possível fazer ainda um agendamento de construções e tarefas, que vão poder ser automatizadas. Tudo feito de uma forma bem simples, as tarefas que forem básicas serão executadas sem nenhum tipo de configuração.

Algumas ferramentas de verificações de erros e também melhoria de desempenho serão conseguidas. Os plug-ins vão permitir que a integração possa ser integrada a construção de projeto sem que causar maiores problemas.

O Hudson é tão completo que o mesmo ainda permite que seja mais fácil de ter informações do acompanhamento da produção. É possível extrair relatórios específicos sobre tudo.

CruiseControl: Para os mais puristas, uma das principais ferramentas de integração contínua é justamente essa. Trata-se de algo que maduro e ainda do tempo em que o Windows XP era sinônimo de ter a mais alta tecnologia. O Pentium 4 então reinava e todos tinham o Fly Simulator no seu monitor de Cristal Líquido.

Trata-se de uma ferramenta que é madura e está fora desse escopo, pois as outras evoluíram bastante. Ainda sim merece ser mencionada,

porque o que marcou história deve sempre ser lembrado por todas as partes.

Essa é uma das soluções mais antigas, portanto, tem aproximadamente 15 anos de lançamento. A ferramenta foi lançada para como uma espécie de quadro e a ideia básica era conseguir controlar os famosos builds.

Não deixando de lado os testes e também ciclos de vida de todo o desenvolvimento de um software. Essa ferramenta tem toda a sua base em Java, ou seja, é até simples de conseguir desenvolver as funções nela.

Como tudo que é bom pode sempre melhorar, ainda existe implementações do Cruise Control em .NET e também rubi. Pois é, é necessário citar essa aqui para aqueles que são “das antigas”, apreciando uma boa e velha nostalgia.

Bamboo: A Atlassian também tem um servir de Integração continue e o mesmo tem o nome de Bamboo. Trata-se de uma ferramenta que é bem eficiente e promete o que ela cumprir, porque trata-se de algo código aberto. Existirá uma parte em que é gratuita e outra que será cobrado, portanto, é preciso ter muita atenção.

As organizações comerciais serão cobradas de acordo com o número de builds que forem necessários. É justamente aqui que será necessário estar atento para descobrir se realmente irá compensar ou não, ou seja, faça as contas.

A ferramenta também fará a utilização de vários plug-ins que prometem fazer a personalização do seu uso. Sem esquecer das suas funcionalidades que são bem próximas das ferramentas que foram citadas anteriormente.

Como trata-se de um produto Atlassian, toda essa integração do JIRA e Clover, a cobertura de código por meio dessa ferramenta, tudo torna-se fácil. O Bambu oferece essa facilidade e talvez por essa razão esteja sendo um dos mais usados.

Essa funcionalidade e muito interessante para pessoas que não trabalhem com muitos builds, pois será cobrado um valor. Acima de qualquer coisa, será preciso fazer as contas e ver se vai compensar, portanto, tenha um pouco de atenção nisso.

Team City: Essa é uma das principais ferramentas de integração contínua e vai conseguir integrar todo o trabalho. Diversas vezes durante o dia, a base de código vai permanecer consistente ao final de cada integração. A principal função é permitir que a execução de testes automáticas possa ser realizada.

Serão criados vários tipos de testes referentes a cada trecho desse código que é desenvolvido, os mesmos ficam armazenados no TeamCity. Dessa forma, toda vez que um desenvolvedor fizer a alteração ou mesmo a criação desses novos códigos, fazendo em seguida uma simulação do que poderia vir a acontecer.

Se existir as falhas, será mais fácil de perceber o que está errado e aí cabe ao desenvolvedor fazer a correção. O programa permanece o mesmo, ou seja, é possível saber antes aquilo que vai dar certo ou mesmo errado.

Existindo erros, a integração é cancelada, porém se estiver certo, a integração é liberada para ser usada. Isso é primordial, porque poupa o retrabalho e melhora ainda mais os resultados, ou seja, adianta todo o trabalho.