INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E
TECNOLOGIA DE SÃO PAULO
CÂMPUS GUARULHOS

AUTORA: BEATRIZ RODRIGUES ALENCAR, CURSANDO PÔS GRADUAÇÃO
ORIENTADOR: THIAGO SCHUMACHER BARCELOS

UM ESTUDO DE CASO: UMA PROPOSTA DE APLICAÇÃO DA INTEGRAÇÃO CONTÍNUA NA PLATAFORMA SALESFORCE

SÃO PAULO
2019

1 INTRODUÇÃO

Com a evolução da tecnologia, as empresas têm tido a necessidade de lançar produtos inovadores com tempo de entrega cada vez mais curtos. Nesse cenário, foi necessário a adoção de metodologias ágeis, incorporando, dentre outras práticas-modelo, a integração contínua.

Este trabalho irá abordar um estudo de caso em uma empresa multinacional do ramo de seguros localizada na cidade de São Paulo, onde é utilizada a plataforma Salesforce para gerenciar seus clientes. A plataforma Salesforce é um software de prateleira focado na gestão de relacionamento com o cliente (CRM), sendo adquirida pela Internet em um plano de pagamento por mês de servidores implantados no local sob um contrato robusto de licenciamento, reduzindo assim os custos de implementação, infraestrutura e manutenções (COMPUTER WORLD UK, 2018).

Devido ao acesso da autora ao campo verificou-se a necessidade de aplicar a prática da integração contínua pois na equipe ocorre diversos problemas como desperdício de tempo e produtividade.

Portanto, o principal objetivo desta pesquisa é desenvolver um estudo de caso de uma proposta de aplicação da integração continua na plataforma Salesforce.

2 REVISÃO DA LITERATURA

Este capítulo tem como objetivo apresentar os conceitos pertinentes empregados neste trabalho. Os conceitos são fundamentais para compreensão, apresentação e desenvolvimento da prática de Integração Contínua (CI).

2.1 INTEGRAÇÃO CONTÍNUA

Integração Contínua é considerada uma pratica de desenvolvimento de software que tem como características a frequente integração das alterações de código fonte feitas pelos desenvolvedores de uma equipe, podendo ser realizada várias vezes ao dia (FOWLER, 2006).

2.1.1 Práticas da Integração Contínua

Segundo (FOWLER, 2006), não existe uma receita fixa para aplicar a integração contínua, mas o mesmo apresenta alguns passos que levam ao sucesso.

2.1.1.1 Repositório de Código

Manter todos os artefatos em um único repositório de código utilizando uma ferramenta de gerenciamento de código fonte, de forma que todos os membros da equipe consigam fazer o check-out do projeto de qualquer lugar ou em qualquer máquina onde o ambiente de desenvolvimento não esteja previamente configurado.

2.1.1.2 Build Automatizada

A obtenção de um sistema executável é um processo complexo pois envolve compilação, manipulação de arquivos, carregamento dos schemas nas bases de dados e assim por diante, entretanto gasta-se um tempo desnecessário fazendo esse trabalho de forma manual. É benéfico que essa construção seja feita de forma automática em um ambiente diferente dos ambientes de desenvolvimento.

2.1.1.3 Build Auto Testável

Incluir os testes automatizados no processo do build traz grandes benefícios como achar bugs não previstos. Caso o teste falhe a build automaticamente deve ser quebrada retornando um feedback para o desenvolvedor, assim tornando o código auto testável.

2.1.1.4 Lançamento de modificações todos os dias

Com a prática de realizar o lançamento das modificações todos os dias é que encontra-se rapidamente os conflitos entre as versões tornando sua solução mais fácil, também melhorando a comunicação dos desenvolvedores verificando as modificações uns dos outros.

O único pré-requisito é que as modificações devem manter o código principal em perfeito estado sendo possível executar perfeitamente o código.

2.1.1.5 Cada commit deve atualizar o repositório principal em uma máquina de integração

O uso do commits diariamente resulta em um sistema em um ótimo estado, eliminando problemas como ambientes diferentes entre as maquinas dos desenvolvedores, falta de atualização das versões, perda de histórico, entre outros.

2.1.1.6 Build Rápida

Manter a build rápida é umas das maiores preocupações na integração contínua, o tempo ideal utilizando métodos ágeis para um *build* é dentro da linha de 10 minutos.

Para não ter problemas com tempo do *build*, é recomendado dividi-lo em dois estágios, sendo alocados no primeiro estágio os testes mais importantes, assim obtendo um feedback instantâneo, encontrando erros com maior agilidade. O segundo estágio tem como objetivo executar todos os testes secundários, assim tendo um *build* completo.

2.1.1.7 Teste em uma cópia do ambiente de produção

Os testes devem ser executados em um ambiente de cópia exata ao ambiente de produção, evitando surpresas na implantação final.

2.1.1.8 Comunicação

O ponto principal da Integração continua é a comunicação, é importante que todos os integrantes da equipe tenham acesso ao estado do software, podendo ser através de um painel digital, e-mails, alertas sonoros, entre vários outros meios, mostrando a todo tempo se houve algum problema no ambiente de integração contínua.

2.1.1.9 Automatize a Implantação do Sistema

Automatizar a implantação do sistema é muito importante para diminuição de erros e horas desperdiçadas dos recursos, com alguns scripts é possível executar o sistema em qualquer ambiente com facilidade, inclusive no ambiente de produção, dando também agilidade em caso de alguma modificação indesejada ou com erros, efetuando rapidamente o *rollback* (reversão).

2.2 GERENCIAMENTO DE RELACIONAMENTO COM O CLIENTE (CRM)

Conforme (SWIFT, 2001), gerenciamento de relacionamento com o cliente é um conjunto de práticas, estratégias de negócio e tecnologias focadas no cliente, utilizadas por empresas e organizações para entender e influenciar o comportamento dos seus clientes. Eles são analisados por meio de comunicações significativas, antecipando suas necessidades, otimizando a rentabilidade e aumentando a lucratividade, permitindo que a empresa seja mais assertiva na captação de novos clientes.

Segundo (KOTLER, 2005), conquistar novos clientes custa de 5 a 7 vezes mais caro do que manter os clientes já existentes. Com isso, utilizar ferramentas como o CRM, permite melhorar a gestão das informações, elaborar as estratégias para aumentar a satisfação e a fidelização dos clientes (OLKOSKI et al., 2009).

2.3 PLATAFORMA SALESFORCE

A *Salesforce* foi fundada em 1999 em San Francisco, pelo ex-executivo da Oracle Marc Benioff, junto com três desenvolvedores de software anteriormente afiliados à consultoria Left Coast Software, Parker Harris, Dave Moellenhoff e Frank Dominguez. A ideia era "tornar o software mais fácil de comprar, mais simples de usar e mais democrático, sem as complexidades de instalação, manutenção e atualizações constantes" (COMPUTER WORLD UK, 2018).

Foi então que nasceu o *Salesforce* no modelo Software como Serviço (Software as a Service - SaaS), sendo um software de gerenciamento de relacionamento com o cliente (CRM) pela Internet em um plano de pagamento por mês de servidores implantados no local sob um contrato robusto de licenciamento, reduzindo assim os custos de implementação, infraestrutura e manutenções (COMPUTER WORLD UK, 2018).

Atualmente a *Salesforce* é uma das empresas tecnológicas mais inovadoras do mundo, pelo fato de ter três atualizações anuais na plataforma disponibilizado para todos os seus clientes novas funcionalidades, serviços e correções (FORBES, 2018).

3 **ESTUDO DE CASO**

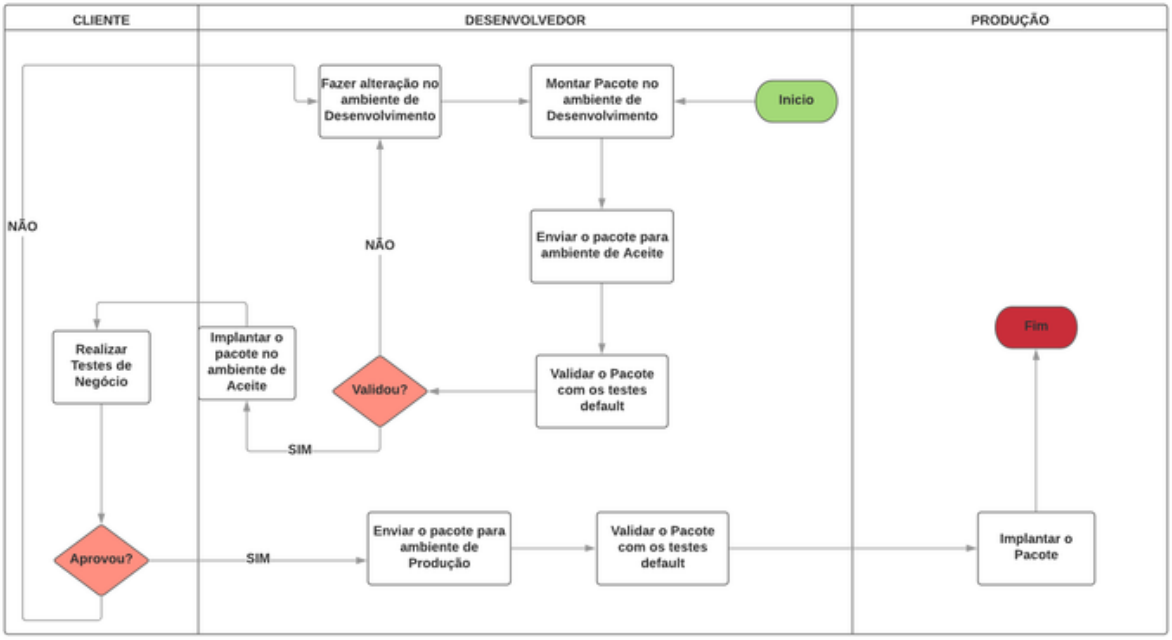
Este trabalho estuda uma empresa multinacional de grande porte. A autora teve acesso ao campo por meio de sua atuação na empresa, que atua no ramo de Seguros com 50.000 colaboradores em mais de 900 escritórios em todo o mundo, atuando na sede do Brasil, onde é realizado o desenvolvimento dos projetos pelo setor de TI.

3.1 PROCESSO ATUAL

A metodologia de trabalho da equipe funciona da seguinte forma. No início de cada ano estão definidos os responsáveis por cada projeto. Todo mês acontece uma reunião de comitê onde as áreas clientes priorizam as correções de bugs ou as melhorias que desejam que seja feito no mês seguinte. Após a reunião é realizado a atribuição das atividades para cada desenvolvedor e realizado a estimativa de esforço para cada atividade, por fim respondendo para o cliente o que exatamente será entregue na GMUD.

GMUD é um dia especifico onde ocorre as publicações do sistema de todo o departamento de TI, a empresa define um calendário anual com todas as datas da GMUD de cada mês sendo sempre de sexta para sábado às iniciando às 18h, a publicação dos pacotes foi ilustrada na imagem diagrama 1 através do diagrama de atividades *swimlanes*.

Diagrama 1 - Processo Atual



Fonte: A autora (2019)

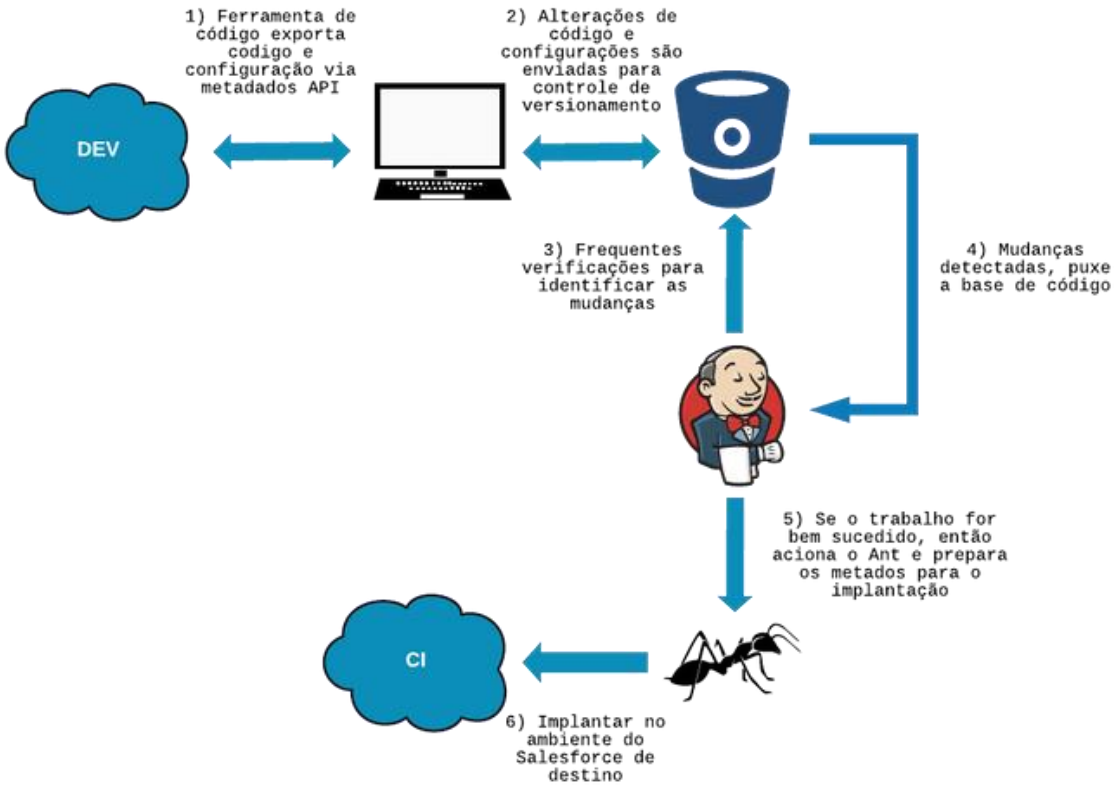
Com este modelo atual foi identificado muitos problemas como perda de código, desperdício de tempo e recursos e falta de comunicação, conforme obtido na aplicação do instrumento de pesquisa o questionário.

3.2 A PROPOSTA

Para solucionar os problemas identificados será proposta a prática da integração contínua, com o objetivo de reduzir o trabalho manual, automatizar a compilação e a criação do pacote.

O estudo foi baseado na proposta publicada por (RUDRAPPA, 2018), abaixo o mesmo apresenta como ocorre o ciclo da prática da integração continua na plataforma *Salesforce* a partir de um ambiente de desenvolvimento.

Figura 1 - Integração Contínua no Salesforce



Fonte: A autora (2019)

Inicialmente, terá que ser feita a padronização de todos os projetos, deixando-os com a mesma estrutura de pastas, assim tornando a forma de compilação, execução dos testes, criação dos pacotes todas iguais.

Para o estudo foi criado a seguinte estrutura:

C:\Users\Beatriz\Documents\GIT\Salesforce

Contendo dentro da pasta do projeto *Salesforce* uma pasta chamada *build*, que haverá todas configurações para a execução *do build* automático.

C:\Users\Beatriz\Documents\GIT\Salesforce\build

Conforme documentação da Salesforce, uma das formas para se fazer o build automático é utilizando o Ant Migration Tool, que é um utilitário de linha de comando baseado em Java/Ant para mover metadados entre um diretório local e uma organização do *Salesforce*. Você pode usar o *Ant Migration Tool* para recuperar componentes, criar uma implantação com *script* e repetir padrões de implantação (Salesforce, 2019a).

Foi criado uma nova pasta chamada *lib* dentro da pasta de *build* para incluir o ant-salesforce.jar, que esta disponível para *download* na própria plataforma do *Salesforce*.

Seguindo a documentação o primeiro passo é criar um arquivo chamado *build* com a extensão *properties* para apontar para uma organização do *Salesforce*, informando nome de usuário, senha, token e url do servidor (Salesforce, 2019b).

Figura 2 - Build.properties

```
# build.properties
#

# Specify the login credentials for the desired Salesforce organization
sf.username = <Insert your Salesforce username here>
sf.password = <Insert your Salesforce password here>
#sf.sessionId = <Insert your Salesforce session id here.  Use this or username/password above.  Cannot use both>
#sf.pkgName = <Insert comma separated package names to be retrieved>
#sf.zipFile = <Insert path of the zipfile to be retrieved>
#sf.metadataType = <Insert metadata type name for which listMetadata or bulkRetrieve operations are to be performed>

# Use 'https://login.salesforce.com' for production or developer edition (the default if not specified).
# Use 'https://test.salesforce.com' for sandbox.
sf.serverurl = https://login.salesforce.com

sf.maxPoll = 20
# If your network requires an HTTP proxy, see http://ant.apache.org/manual/proxy.html for configuration.
#
```

Fonte: A autora (2019)

Segundo passo é criar um outro arquivo chamado *build.xml*, este arquivo será utilizado para parametrizar uma series de comandos a serem executados pelo Ant (Salesforce, 2019c).

Figura 3 - Build.xml

```
<project name="Sample usage of Salesforce Ant tasks" default="test" basedir="." xmlns:sf="antlib:com.salesforce">

    <property file="build.properties"/>
    <property environment="env"/>

    <taskdef resource="com/salesforce/antlib.xml" uri="antlib:com.salesforce">
        <classpath>
            <pathelement location="/lib/ant-salesforce.jar" />
        </classpath>
    </taskdef>

    <!-- Shows deploying code & running tests for code in directory -->
    <target name="deployCode">
      <sf:deploy username="${sf.username}"
                 password="${sf.password}"
                 sessionId="${sf.sessionId}"
                 serverurl="${sf.serverurl}"
                 maxPoll="${sf.maxPoll}"
                 deployRoot="C:\Users\Beatriz\Documents\GIT\Salesforce\src"
                 testLevel="RunAllTestsInOrg"
                 rollbackOnError="true">
      </sf:deploy>
    </target>

    <!-- Shows check only; never actually saves to the server -->
    <target name="deployCodeCheckOnly">
      <sf:deploy    username="${sf.username}"
                 password="${sf.password}"
                 sessionId="${sf.sessionId}"
                 serverurl="${sf.serverurl}"
                 maxPoll="${sf.maxPoll}"
                 deployRoot="C:\Users\Beatriz\Documents\GIT\Salesforce\src"
                 checkOnly="true"/>
    </target>
```

Fonte: A autora (2019)

Terceiro passo criar um arquivo chamado *package.xml* na pasta do projeto onde se encontra todos os componentes, para listar todos os componentes que deseja recuperar ou implantar em uma unica solicitação (Salesforce, 2019d).

No estudo será criado o arquivo na pasta *src*:

C:\Users\Beatriz\Documents\GIT\Salesforce\src

Figura 4 - Package.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>*</members>
        <name>ApexClass</name>
    </types>
    <types>
        <members>*</members>
        <name>ApexTrigger</name>
    </types>
    <version>45.0</version>
</Package>
```

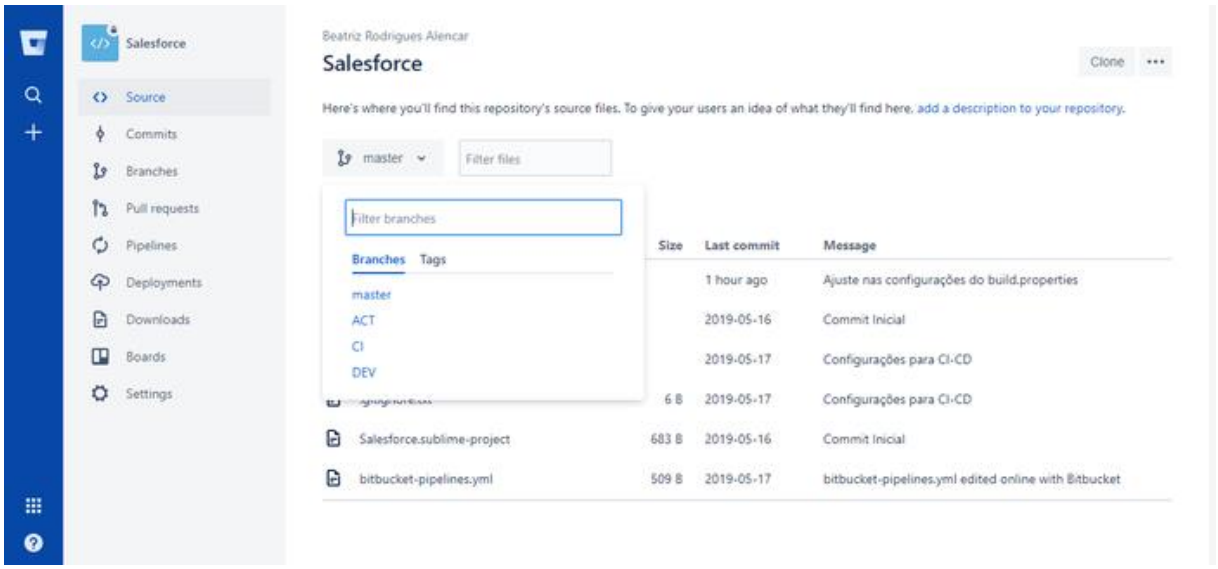
Fonte: A autora (2019)

Para tal, é proposta a migração do código para o repositório *BITBUCKET*, de forma a permitir o controle de versionamento, autor e data/hora de cada *check-in* e merge (Bitbucket, 2019).

Foi criado um repositório chamado *Salesforce*, com as seguintes *branches*:

1. Master - Ambiente de produção.
2. ACT - Clonado a partir da master, especifico para os testes do cliente.
3. CI - Clonado a partir da master, especifico para o merge e testes integrados.
4. DEV - Clonado a partir da master, especifico para o desenvolvimento.

Figura 5 - Repositório Salesforce - Bitbucket

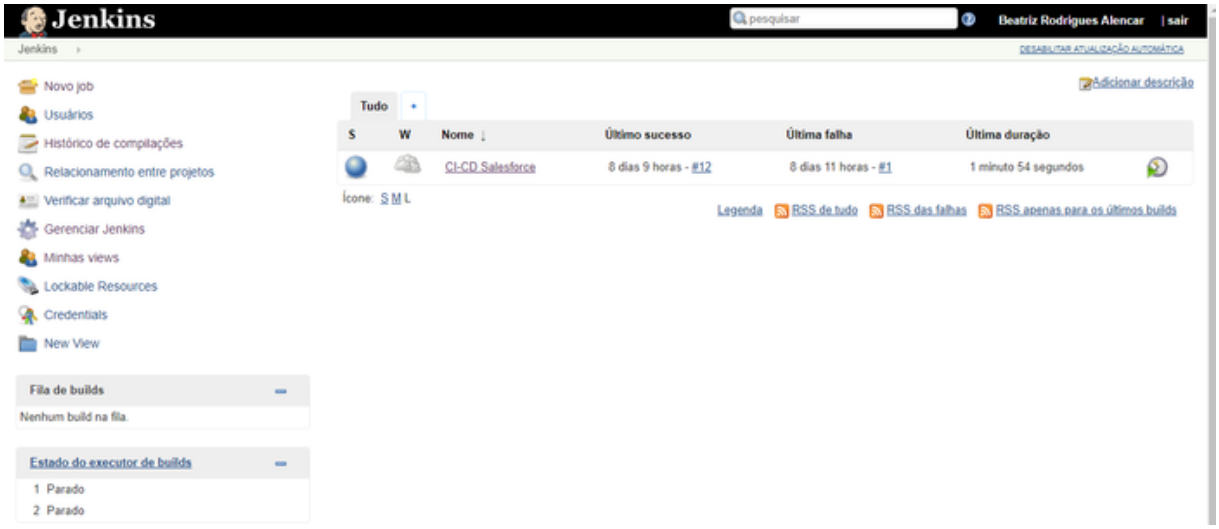


Fonte: A autora (2019)

Para a automação e execução dos *build's* será utilizado o *Jenkins*, servidor de automação de código aberto mantido por uma comunidade grande de desenvolvedores, testers, designers e pessoas interessadas no assunto (JENKINS, 2019).

Foi criado um novo *job* do tipo *free-style* com nome CI-CD *Salesforce* para fazer as devidas configurações.

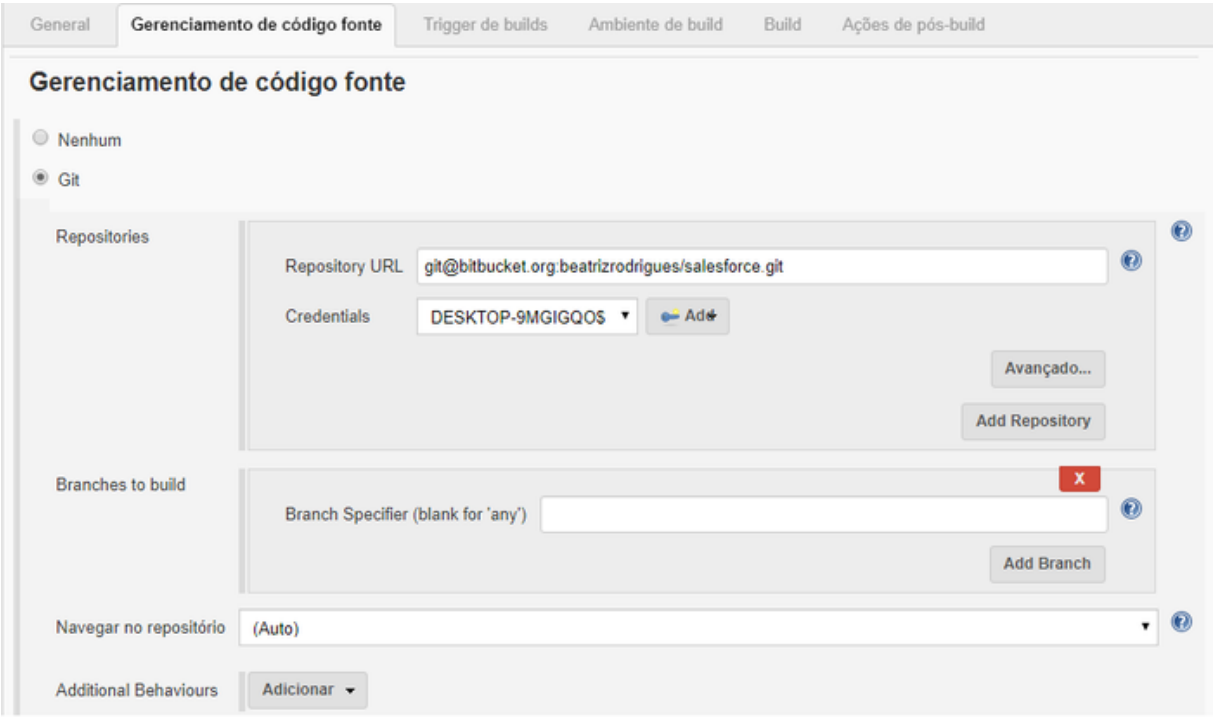
Figura 6 - Jenkins



Fonte: A autora (2019)

Primeira etapa a ser realizada é a escolha do gerenciamento de código fonte, conforme a proposta é o *Git*. É preciso informar as credenciais para fazer o vínculo com o *Jenkins*.

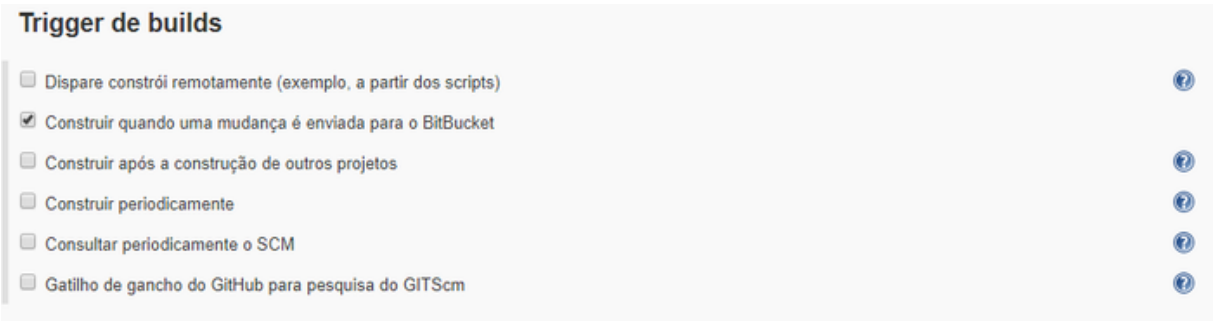
Figura 7 - Jenkins - Gerenciamento de Código Fonte



Fonte: A autora (2019)

Segunda etapa é realizada para informar como será o disparo da *build*, a autora escolheu o disparo feito quando é realizado uma mudança no *Bitbucket* conforme figura 8.

Figura 8 - Jenkins - Triggers de Builds



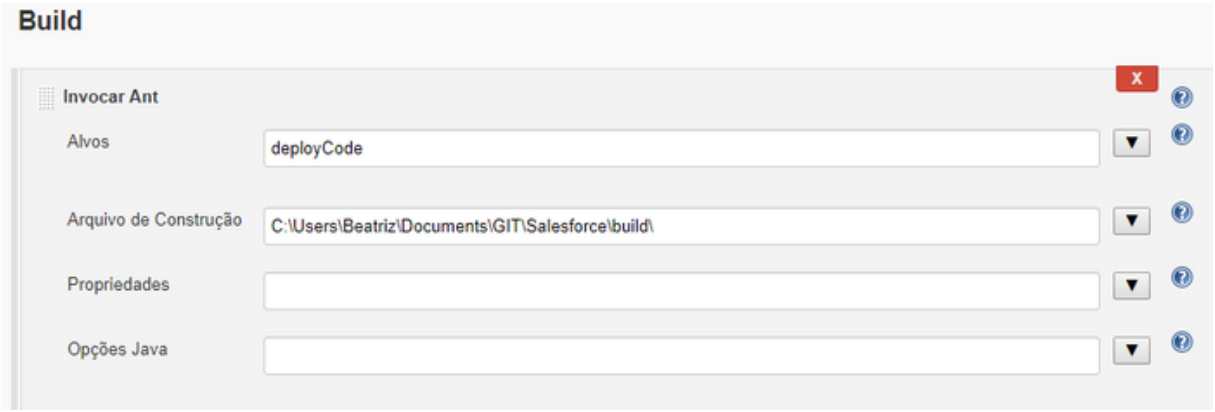
Fonte: A autora (2019)

A configuração para execução do *build* é preciso selecionar o *Ant* e preencher os campos solicitados:

1. Informar o nome da *Target* do arquivo *build.xml* para ser executada.

2. Informar o caminho onde se encontra os arquivos de *build.xml* e *build.properties* do projeto.

Figura 9 - Jenkins - Build



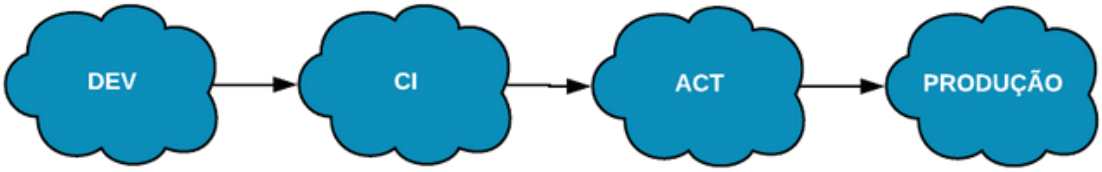
Fonte: A autora (2019)

Na plataforma *Salesforce* é possível criar vários ambientes a partir da organização de produção de acordo com a licença comprada. Existe vários tipos de ambientes dependendo da necessidade.

No estudo proposto utilizamos a seguinte estrutura de ambientes para aplicar a integração contínua, ilustrado na figura 10.

- Ambiente de DEV do tipo *Developer Sandbox*, pois será utilizado somente por desenvolvedores e só será copiado os metadados.
- Ambiente de CI do tipo *Sandbox* de Cópia Parcial, destinada a ser usada como ambiente de teste.
- Ambiente de ACT do tipo *Sandbox* Completo, é uma réplica da organização de produção, incluindo todos os dados, como registros de objeto e anexos, e metadados.
- Produção é a organização atual, onde se encontra o seu sistema final (Salesforce, 2019e).

Figura 10 - Estrutura de Ambiente - Salesforce



Fonte: A autora (2019)

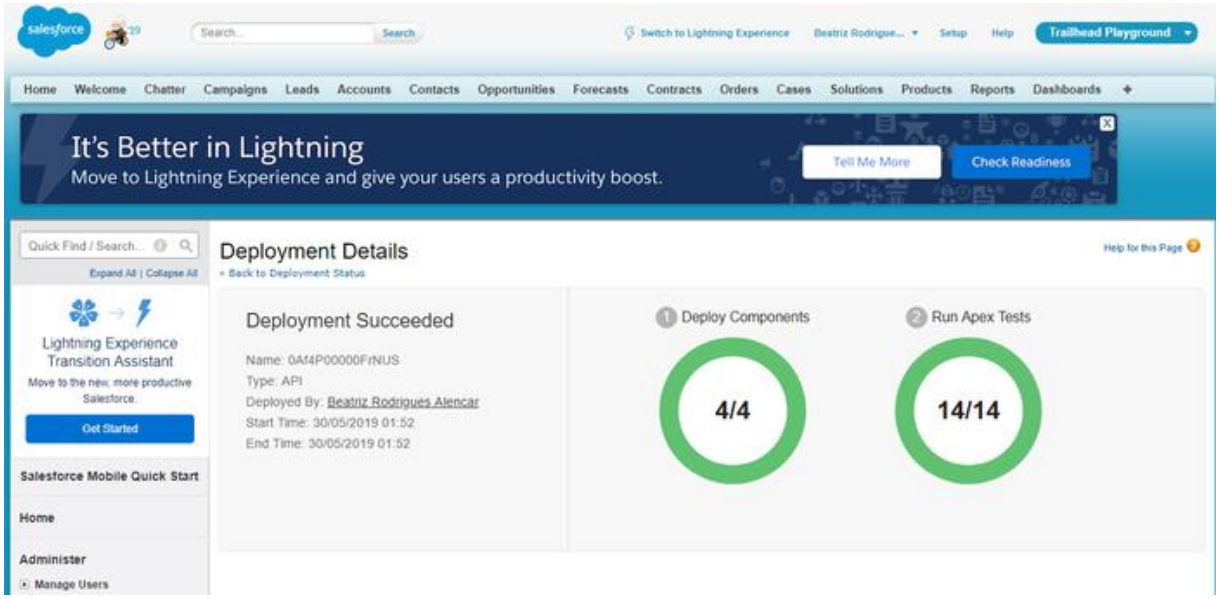
Realizando todas as etapas, a autora colocou em prática o estudo, fez um *commit* na *branch* de desenvolvimento e o *merge* para *branch* de CI, assim o *Jenkins* identificou automaticamente executando o *build Ant e* implantando no ambiente de CI do *Salesforce*, podendo ser visualizado no log do servidor do *Jenkins* conforme figura 11 ou no status da plataforma *Salesforce* conforme figura 12.

Figura 11 - Jenkins - Log do Build

```
01:54:30 [sf:deploy] 01:52:46.311 (1462882873)|SOQL_EXECUTE_BEGIN|[51]|Aggregations:0|SELECT
UserPreferencesLightningExperiencePreferred FROM User WHERE Id = :tmpVar1
01:54:30 [sf:deploy] 01:52:46.311 (1509237722)|SOQL_EXECUTE_END|[51]|Rows:1
01:54:30 [sf:deploy] 01:52:46.311 (1509534258)|ENTERING_MANAGED_PKG|th_con_app
01:54:30 [sf:deploy] 01:52:46.311 (1509980322)|SOQL_EXECUTE_BEGIN|[67]|Aggregations:0|SELECT StartUrl FROM AppMenuItem WHERE
(NamespacePrefix = 'th_con_app' AND Name = 'Trailhead_Playground')
01:54:30 [sf:deploy] 01:52:46.311 (1538305458)|SOQL_EXECUTE_END|[67]|Rows:1
01:54:30 [sf:deploy] 01:52:46.601 (1601375087)|USER_INFO|[EXTERNAL]|0054P00000Z2eB|beatriz_rodrigues3@curious-goat-s9aoqp.com|(GMT-
03:00) Brasilia Time (America/Sao_Paulo)|GMT-03:00
01:54:30 [sf:deploy] 01:52:46.601 (1601417863)|ENTERING_MANAGED_PKG|th_con_app
01:54:30 [sf:deploy] 01:52:46.601 (1601706478)|ENTERING_MANAGED_PKG|th_con_app
01:54:30 [sf:deploy] 01:52:46.601 (1601818818)|ENTERING_MANAGED_PKG|th_con_app
01:54:30 [sf:deploy] 01:52:46.601 (1602208001)|SOQL_EXECUTE_BEGIN|[20]|Aggregations:0|SELECT id FROM Profile WHERE Name = 'System
Administrator'
01:54:30 [sf:deploy] 01:52:46.601 (1606100321)|SOQL_EXECUTE_END|[20]|Rows:1
01:54:30 [sf:deploy] 01:52:46.601 (1606533616)|ENTERING_MANAGED_PKG|th_con_app
01:54:30 [sf:deploy] 01:52:46.601 (1607809666)|DML_BEGIN|[13]|Op:Insert|Type:User|Rows:1
01:54:30 [sf:deploy] 01:52:46.601 (1710627043)|DML_END|[13]
01:54:30 [sf:deploy] 01:52:46.601 (1730306887)|ENTERING_MANAGED_PKG|th_con_app
01:54:30 [sf:deploy] 01:52:46.601 (1730343101)|ENTERING_MANAGED_PKG|th_con_app
01:54:30 [sf:deploy] 01:52:46.601 (1730785494)|SOQL_EXECUTE_BEGIN|[51]|Aggregations:0|SELECT
UserPreferencesLightningExperiencePreferred FROM User WHERE Id = :tmpVar1
01:54:30 [sf:deploy] 01:52:46.601 (1741975672)|SOQL_EXECUTE_END|[51]|Rows:1
01:54:30 [sf:deploy] 01:52:46.601 (1742130201)|ENTERING_MANAGED_PKG|th_con_app
01:54:30 [sf:deploy] ********** DEPLOYMENT SUCCEEDED **********
01:54:30 [sf:deploy] Finished request 0Af4P00000FrNUSSA3 successfully.
01:54:30
01:54:30 BUILD SUCCESSFUL
01:54:30 Total time: 1 minute 50 seconds
01:54:30 Finished: SUCCESS
```

Fonte: A autora (2019)

Figura 12 - Salesforce - Deploy



Fonte: A autora (2019)

Na figura 12, no status da implantação direto na plataforma Salesforce, podemos verificar que o primeiro círculo indica a quantidade de componentes enviados e se a compilação foi bem-sucedida, tornando o círculo em verde. O segundo círculo indica a execução dos testes unitários (SOUSA, 2017), também se for bem-sucedida o círculo ficará verde, e se ocorrer alguma falha o círculo ficará em vermelho e abaixo a descrição dos problemas ocorridos (Salesforce, 2019f).

3.3 RESULTADOS ESPERADOS

Como resultados a prática da integração contínua na plataforma *Salesforce* demonstra-se muito eficaz conforme proposta apresentada na seção 3.3, a automatização do processo fez com que os trabalhos manuais como a montagem de pacote, desperdício de horas dos recursos conforme problemas apontados pelos participantes no questionário mencionado na seção 3.1, foram sanados com a utilização da ferramenta *Jenkins*.

Na apresentação do problema de comunicação apresentado pela autora na seção 3.1, pode-se afirmar que tem uma melhora significativa pois a ferramenta *Jenkins* também mostra o feedback no console da própria ferramenta apresentado na figura 17 da seção 3.3 e o envio por e-mail de forma automática pela mesma.

No caso da validação dos testes, conforme apresentado na seção 2.1 o (FOWLER, 2006) menciona que na maioria dos projetos que utilizam métodos ágeis como

XP atingi o tempo ideal de 10 minutos, porém na plataforma *Salesforce* em um projeto considerado grande com mais de 500 classes, os testes podem demorar mais de 30 minutos.

Nesse caso na visão da autora a execução dos testes com a prática da integração contínua não implica, pois cada projeto tem sua arquitetura diferente podendo aumentar ou diminuir o tempo de execução dos testes.

4 **CONCLUSÃO**

Não foi realizada a implantação de fato na empresa por questões burocráticas, foi utilizado uma conta de desenvolvimento na plataforma *Salesforce* da própria autora para ser realizado o estudo, porém podemos afirmar que os resultados são os mesmos, pois só muda a quantidade de informações que tem no ambiente, com tudo conclui-se que a proposta trará grandes benefícios para a equipe conforme resultados apresentados.

REFERÊNCIAS

BITBUCKET. Version control software for professional teams. **Bitbucket**. 2019. Disponível em: <https://bitbucket.org/product/version-control-software>. Acesso em: 28 mai. 2019.

COMPUTER WORLD UK. A brief history of Salesforce.com. **COMPUTER WORLD UK**. 2018. Disponível em: <https://www.computerworlduk.com/cloud-computing/brief-history-of-salesforcecom-3678095>. Acesso em: 7 fev. 2019.

FORBES. 10 empresas mais inovadoras do mundo em 2018. **Forbes**. 2018. Disponível em: <https://forbes.uol.com.br/listas/2018/06/10-empresas-mais-inovadoras-do-mundo-em-2018>. Acesso em: 7 fev. 2019.

FOWLER, M. **Continuos Integration**. 2006. Disponível em: <https://martinfowler.com/articles/continuousIntegration.html>. Acesso em: 20 set. 2018.

JENKINS. About Jenkins. **JENKINS**. 2019. Disponível em: <https://jenkins.io/press/#about-jenkins>. Acesso em: 22 mai. 2019.

KOTLER, P. **Administração de Marketing**. Pearson Prentice Hall, 2005.

OLKOSKI, G et al. **Marketing de Relacionamento e Software de CRM: estudo de caso em uma concessionária de automóveis**. Santa Maria: Revista de Adminstração da UFSM, 2009. Disponível em: <https://periodicos.ufsm.br/reaufsm/article/view/1638/926>. Acesso em: 3 mai. 2019.

RUDRAPPA, V. Salesforce Continuous Integration (CI/CD). **Linkedin**. 2018. Disponível em: <https://www.linkedin.com/pulse/salesforce-continuous-integration-cicd-vivek-rudrappa/>. Acesso em: 28 mai. 2019.

SALESFORCE. Ant Migration Tool Guide. **Developer Salesforce**. 2019a. Disponível em: <https://developer.salesforce.com/docs/atlas.en-us.daas.meta/daas/forcemigrationtool.htm>. Acesso em: 26 mai. 2019.

SALESFORCE. Constructing a Project Manifest. **Developer Salesforce**. 2019d. Disponível em: <https://developer.salesforce.com/docs/atlas.en-us.daas.meta/daas/daas_package.htm>. Acesso em: 26 mai. 2019.

SALESFORCE. Creating Retrieve Targets. **Developer Salesforce**. 2019c. Disponível em: <https://developer.salesforce.com/docs/atlas.enus.daas.meta/daas/forcemigrationtool_build.htm>. Acesso em: 3 jun. 2019.

SALESFORCE. Entering Salesforce Connection Information. **Developer Salesforce**. 2019b. Disponível em: <https://developer.salesforce.com/docs/atlas.en-us.daas.meta/daas/forcemigrationtool_connect.htm>. Acesso em: 26 mai. 2019.

SALESFORCE. Monitor Deployments. **Help Salesforce**. 2019f. Disponível em: <https://help.salesforce.com/articleView?id=deploy_monitoring.htm>. Acesso em: 31 mai. 2019.

SALESFORCE. Sandbox Types and Templates. **Help Salesforce**. 2019e. Disponível em: <https://help.salesforce.com/articleView?id=create_test_instance.htm>. Acesso em: 28 mai. 2019.

SOUSA, F. Entendendo os Testes Unitários. **Sou Force**. 2017. Disponível em: <https://souforce.cloud/entendendo-os-testes-unitarios/>. Acesso em: 30 mai. 2019.

SWIFT, R. **CRM, Customer Relationship Management: O Revolucionário Marketing de Relacionamento Com o Cliente**. 13. ed. Rio de Janeiro: Elsevier, 2001.