# Automação de *build* de servidores utilizando o *Red Hat Ansible*

| | |
|---|---|
| **Elaborador:** | Vitor Henrique de Souza |
| **Orientador:** | Marcus Vinícius Lahr Giraldi |

**Faculdade de Tecnologia de Americana – Ministro Ralph Biasi**

**FICHA CATALOGRÁFICA – Biblioteca Fatec Americana - CEETEPS**
**Dados Internacionais de Catalogação-na-fonte**

S719a SOUZA, Vitor Henrique de

Automação de build de servidores utilizando o Red Hat Ansible. / Vitor Henrique de Souza. – Americana, 2019.

23f.

Monografia (Curso Superior de Tecnologia em Segurança da Informação) - - Faculdade de Tecnologia de Americana – Centro Estadual de Educação Tecnológica Paula Souza

Orientador: Prof. Esp. Marcus Vinícius Lahr Giraldi

1 Linux – sistema operacional I. GIRALDI, Marcus Vinícius Lahr II. Centro Estadual de Educação Tecnológica Paula Souza – Faculdade de Tecnologia de Americana

CDU: 681.3.066

VITOR HENRIQUE DE SOUZA

**Automação de *build* de servidores utilizando o *Red Hat Ansible Automation***

Trabalho de graduação apresentado como exigência parcial para obtenção do título de Tecnólogo em Segurança da Informação pelo CEETEPS/Faculdade de Tecnologia – FATEC/ Americana.

Área de concentração: Segurança da Informação

Americana, 7 de dezembro de 2019.

**Banca Examinadora:**

Marcus Vinícius Lahr Giraldi
Especialista
Fatec Americana

Eduardo Antonio Vicentini
Mestre
Fatec Americana

Edson Roberto Gaseta
Mestre
Fatec Americana

FACULDADE DE TECNOLOGIA DE AMERICANA

# SUMÁRIO

**Lista de figuras**



**Lista de tabelas**

# 1 Objetivo deste documento

Com o crescimento das redes e das tecnologias referentes à tecnologia da informação, vem se tornando cada vez mais comum a utilização de ferramentas de automação para se garantir *softwares* mais estáveis e escaláveis. Para isso, foram criados diversos *frameworks* de automação, como *Puppet, Chef, IBM Bigfix* e também o que será utilizado aqui: O *Red Hat Ansible*.

Essas ferramentas surgiram através da necessidade estampada em um conceito que vem sendo extremamente utilizado nos ambientes de TI: O *DevOps*, que é uma unificação de *Development* (Desenvolvimento) e *Operations* (Operações). Com a criação do *DevOps*, acabou-se o conflito entre as duas frentes, tornando mais ágil a implementação de *softwares*.

No dia a dia de qualquer pessoa que trabalha na área de suporte de TI, existem as tarefas repetitivas e longas, inclusive na *IBM*. Foi utilizada a ferramenta *Ansible* para automação de diversas atividades, envolvendo *build* de servidores.

Este relatório visa mostrar as vantagens de se automatizar processos de TI, evidenciando os ganhos e explicando as ferramentas. Mais especificamente, será mostrado como o *Ansible* pôde ser a opção mais viável para o caso exposto aqui, e também o tempo ganho com a implementação dele.

# 2 Fundamentação teórica

Durante o documento, diversos temas envolvendo a solução serão abordados. Neste capítulo, serão explicados conceitos que ajudarão no entendimento do relatório como um todo.

## 2.1 *Open source*

*Softwares open source* são programas que possuem o código livre para qualquer pessoa utilizar ou modificar. Para Fuggetta (2003), pode ser considerado como código aberto "[...] qualquer *software* cuja licença garanta ao seu usuário liberdade relacionada ao uso, alteração e redistribuição. Seu aspecto fundamental é o fato de o código-fonte estar livremente disponível para ser lido, estudado ou modificado por qualquer pessoa"

Sobre o desenvolvimento de um projeto *open source*, vale lembrar que todos os que trabalham nele o fazem de maneira voluntária, buscando motivações que não o dinheiro para completar as tarefas que os mesmos escolheram executar.

Reis (2003) classifica as pessoas que contribuem para os projetos *open source* da seguinte forma:

- Usuários:
  - Não participantes - Apenas utilizam o *software*, sem dar quaisquer *feedbacks* mesmo quando encontram *bugs*, não contribuindo assim para a sua melhoria;
  - Participantes - Além de utilizar o programa, ainda contribuem com análises, informações de erros e discussões sobre possíveis melhorias.
- Desenvolvedores:
  - Esporádicos - Desenvolvedores que contribuem esporadicamente em um projeto, geralmente sendo responsáveis por resolver pequenos problemas;
  - Ativos - Contribuem assiduamente para o desenvolvimento do *software*, normalmente programando as partes mais complexas.
- Equipe de coordenação - Time responsável pelo gerenciamento do projeto como um todo, estabelecendo cronogramas e entrada de colaboradores na equipe.

## 2.2 Automação

A *Red Hat* ([s. d.]) define da seguinte forma o que é automação em seu próprio *site*:

> "[...] é o uso de *software* para criar instruções e processos reproduzíveis capazes de substituir ou reduzir a interação humana com os sistemas de TI. O *software* de automação trabalha dentro dos limites dessas instruções, ferramentas e estruturas para executar tarefas com pouca ou nenhuma intervenção humana."

Também pode ser chamada de 'Automação de infraestrutura', e na teoria, deve abranger qualquer processo de TI de alguma forma (*RED HAT*, [s. d.]).

Ainda segundo a *Red Hat* ([s. d.]), "[...] Ambientes de TI modernos e dinâmicos precisam ter a capacidade de escalar em uma velocidade jamais vista, e a automação de TI é fundamental para tornar isso possível."

Ao implementar a automação em TI, é possível eliminar processos cansativos e manuais que acabam por gastar o tempo do time, permitindo assim que o mesmo se torne mais produtivo. Também reduz erros e libera tempo que pode ser focado em inovações (*RED HAT*, [s. d.]).

## 2.3 *Ansible Automation*

Segundo a *Red Hat* ([s. d.]), "[...] O *Ansible Automation* é o *software* da *Red Hat* criado para automatizar o provisionamento, a configuração, o gerenciamento e a implantação."

Em outras palavras, o *Ansible* é considerado uma ferramenta *open source* de gerenciamento de configuração, assim como outras do mercado (*Chef, Puppet* e *SaltStack*, por exemplo).

Para a implementação do *Ansible*, é necessário um servidor ou *container* que tenha o *Python* funcional e o *Ansible* instalado, e um usuário que tenha acesso aos servidores que serão gerenciados.

Para explicar melhor o que é o *Ansible*, é preciso esclarecer alguns pontos e características do mesmo.

### 2.3.1 *Agentless*

Segundo a própria *Red Hat* ([s. d.]), "[...] O *Ansible Automation* não requer agentes. Por isso, não é necessário instalar *software* nos componentes que você quer automatizar."

Para Maes (1996), agentes são “[...] Um sistema computacional que habita um dado ambiente, sente e age nesse ambiente, e ao fazê-lo realiza um conjunto de objetivos ou tarefas para o qual foi projetado”.

Ou seja, com aplicações que utilizam agentes (como o *Chef*, por exemplo), cada servidor a ser gerenciado precisa ter um agente instalado para que a automação aconteça. Felizmente, não é o caso do *Ansible*, por se tratar de uma ferramenta *agentless* (Que não precisa de agente para o funcionamento), tornando assim, a sua implementação mais simples e rápida.

### 2.3.2 Conectividade

O *Ansible* se conecta em seus *hosts* através de conexão *SSH*, se for um servidor *Linux*, ou *RDP*, se for *Windows*. Apenas servidores *Linux* foram incluídos nessa inovação, sendo assim, o foco será na conectividade deles.

Para a Cisco ([s. d.]), "[...] O *Secure Shell* (*SSH*) é um protocolo que fornece uma conexão segura de acesso remoto aos dispositivos de rede.". Sendo assim, pode-se considerar a conexão utilizada pelo *Ansible* como segura.

Quanto a forma de se utilizar a conexão *SSH* com o *Ansible*, existem basicamente três formas:

- Conexão com senha simples, passando a senha como argumento ao executar.
- Conexão com senha criptografada, onde apenas um texto criptografado é passado como argumento.
- Conexão via chave *SSH*, que é quando se gera uma chave no servidor gerenciado em formato de arquivo e o transfere para o servidor onde o *Ansible* está instalado.

  O mais utilizado costuma ser a chave *SSH*, devido à sua segurança e praticidade.

### 2.3.3 Módulos

Módulos são pequenos programas escritos em *Python* que já vêm instalados por padrão quando se implementa o *Ansible*. Segundo a *Red Hat* ([s. d.], tradução nossa), são "[...] centenas de módulos para que os desenvolvedores, engenheiros, equipes pequenas e escritores de *playbooks* automatizem as atividades diárias."

A *Red Hat* ([s. d.], tradução nossa) apresenta a seguinte definição para os módulos do *Ansible*:

> "[...] O *Ansible* entrega todos os módulos para os sistemas remotos e executa as tarefas conforme solicitado, para implementar a configuração desejada. Estes módulos são executados com credenciais fornecidas pelo usuário, incluindo suporte para o *sudo* e até mesmo *Kerberos*, e limpa os

> mesmos quando estes se completam. O *Ansible* não precisa de privilégios de *root* de *login*, chaves *SSH* específicas ou usuários dedicados e respeita o modelo de segurança do sistema sob gerenciamento."

### 2.3.4 *Playbooks*

*Playbooks* são conjuntos de tarefas que serão executadas nos servidores gerenciados pelo *Ansible*, em que essas tarefas chamam a execução de módulos *Ansible*.

A *Red Hat* ([s. d.], tradução nossa) explica que:

> "[...] Em um nível básico, os *playbooks* podem ser usados para gerenciar configurações e implantações em máquinas remotas. Em um nível mais avançado, eles podem sequenciar lançamentos de várias camadas que envolvem atualizações contínuas e delegar ações a outros *hosts*, interagindo com servidores de monitoramento e balanceadores de carga ao longo do caminho."

Esses *playbooks* são escritos na linguagem de marcação *YAML*, tendo um mínimo de sintaxe. Foi criado dessa forma para não ser nem parecer com linguagens de *script* (*RED HAT*, [s. d.]), sendo assim mais fácil para não-programadores entenderem o que está sendo executado pelo *playbook*.

Os desenvolvedores preferiram a utilização do *YAML* por se tratar de uma linguagem de marcação que utiliza listas (Conforme figura 1.), o que torna muito mais fácil a compreensão do que linguagens como *HTML* e *XML*, que possuem uma sintaxe mais poluída (*RED HAT*, [s. d.]).

**Figura 1: Exemplo simples de *playbook***

```
---
-  name: install and start Apache webserver
   hosts: webservers
   user: root

   tasks:
      name: install httpd
      yum: name=httpd state=present

      name: start httpd
      service: name=httpd state=running
```

**Fonte: Harbinger Systems**[1]

## 2.4 *Jenkins*

*Jenkins* é um *software open source* criado para ser utilizado como integração contínua e entrega contínua. Segundo o próprio *Jenkins* ([s. d.], tradução nossa) em seu *site*, "[...] *Jenkins* é um servidor de automação de código aberto independente, que pode ser usado para automatizar todos os tipos de tarefas relacionadas à construção, teste, entrega ou desenvolvimento de *softwares*"

O *Jenkins* é extremamente simples de ser implementado, visto que "[...] pode ser instalado através de pacotes nativos de sistema, *Docker*, ou até mesmo rodando de forma autônoma em uma máquina com o *Java Runtime Environment* (*JRE*) instalado." (*JENKINS*, [s. d.], tradução nossa).

Para utilizar o *Jenkins*, se criam *jobs*, que são tarefas que executam um *script*, comando ou rotina no servidor da aplicação. Esses, podem ser parametrizados, permitindo assim a criação de formulários que o usuário preenche, e os dados são usados como variáveis durante a execução deste *job*, conforme figura 2.

**Figura 2 - Interface no *Jenkins* que o usuário preenche e em seguida executa o *playbook*.**


[1] Disponível em: <https://blog.harbinger-systems.com/wp-content/uploads/2015/03/Install-Apache-Web-server-in-Ansible1.png>. Acesso em: 23 nov. 2019

**Fonte: Autor**

Existem ainda centenas de *plugins* que podem ser adicionados ao *Jenkins* para ter funcionalidades a mais no aplicativo. *Plugins* também podem ser desenvolvidos por qualquer pessoa, e compartilhada na comunidade do programa (*JENKINS*, [s. d.]).

# 3 Ambiente pré-implementação

Neste capítulo, será abordado como o ambiente do cliente estava antes de a automação ser implementada.

## 3.1 Infraestrutura do cliente

No momento, a infraestrutura do cliente conta com dois tipos de servidores *Linux* com relação a seus sistemas operacionais: *Red Hat Enterprise Linux* 7.5 e *Suse Linux Enterprise Server* 12SP3.

Também possui servidores com *Windows Server*, nas versões 2008, 2012 e 2016, porém estes não serão o foco, visto que a solução apenas engloba servidores *Linux*.

O cliente possui dois ambientes aos quais estes servidores pertencem: *PROD* (*Production*) e *QA* (*Quality Assurance*). Servidores de *PROD* são aqueles que estão em produção, sendo diretamente responsáveis pelo lucro operacional. Quanto aos servidores de *QA*, existem dois tipos: Aqueles que fazem parte do processo do cliente, porém não afetam diretamente o lucro, e também os servidores de desenvolvimento, que são utilizados apenas para testes.

Existem aproximadamente 2000 servidores distribuídos entre *PROD* e *QA* atualmente.

## 3.2 Processos de *build*

Para o time de *Linux* da *IBM*, *build* são todos os passos que precisam ser realizados antes de o servidor entrar em produção, como instalação e configuração de *softwares*, configurações de rede e de sistema operacional no servidor.

Abaixo, os processos de *build* realizados manualmente pelo time de *Linux* antes da implementação da automação:

- Checar o *timezone*:

- Para checar, utiliza-se o comando *timedatectl*. O horário deve estar setado como *America/New_York*;
- Se necessário alterar, executar o comando *timedatectl set-timezone America/New_York.*

- Checar os serviços NTP e Chrony:
  - Servidores com Suse precisam ter o serviço NTP instalado;
  - Servidores com RedHat precisam ter o serviço Chrony instalado;
  - Para checar os serviços, utiliza-se o comando *systemctl status* <serviço>.
    - Caso o respectivo serviço não esteja instalado, instalar o pacote com o comando *yum install <serviço>* para RedHat, e *zypper install* <serviço>, para Suse.
- Checar o serviço Syslog:
  - Checar se o serviço está rodando, com o comando *systemctl status* <serviço>
  - Verificar se a *log* está sendo preenchida com o comando *cat /etc/rsyslog.conf | grep *.debug*
- Checar os volumes lógicos:
  - Checar se há disponível um grupo de volume chamado VG02, e que ele esteja em um disco de pelo menos 25 *gigabytes* de tamanho total (comando *lsblk*);
  - Caso tenha o VG02, mas o tamanho do disco seja de apenas 20 *gigabytes* aumentar espaço na console da Amazon (AWS) antes de continuar;
  - Caso não tenha o VG02, e tenha um disco maior que 25 *gigabytes*, criar o VG (*Virtual Group*) com os seguintes comandos: *pvcreate /dev/hd_name* e *vgcreate vg02 /dev/hd_name;*
  - Depois, criar os volumes lógicos no VG02 com o comando *lvcreate*;
  - Adicionar as entradas no arquivo */etc/fstab* para que os volumes lógicos sejam montados automaticamente após algum *boot*.
  - Criar diretórios das aplicações do servidor com o comando *mkdir.*
  - Montar tudo com o comando *mount -a.*
- Preencher o arquivo */etc/sudoers* com o conteúdo necessário.

- Exportação dos pacotes, instalação e configuração das seguintes aplicações através do pacote correspondente:
    - IEM;
    - SEA;
    - SRM;
    - TSM;
    - ITM;
    - Centrify.
- Atualizar o sistema operacional do servidor:
    - Fazer *backup* de arquivos de configuração alterados;
    - Atualizar o sistema operacional e reiniciar a máquina;
    - Checar se houve alteração nos arquivos;
    - Caso tenham sido alterados, voltar a configuração com o *backup*.

### 3.2.1 Tempo gasto no *build*

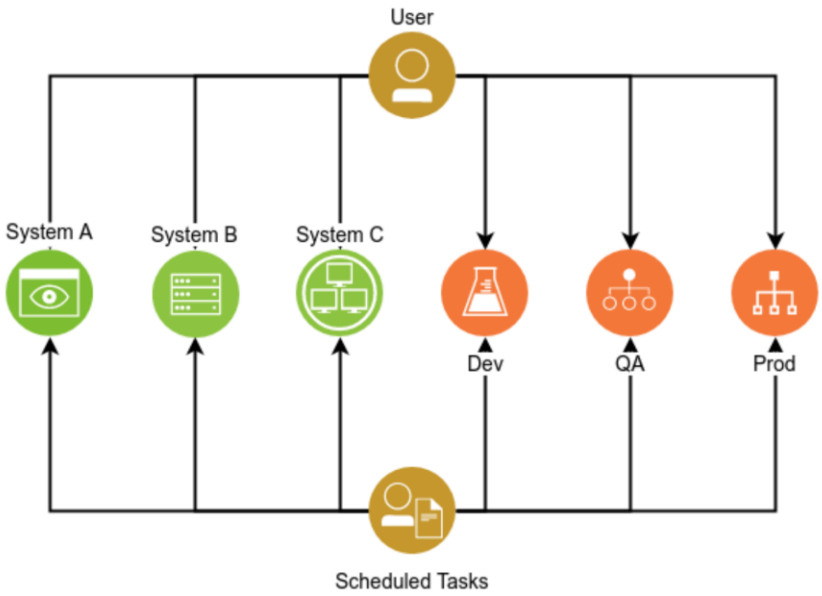
Segundo informado pelo time de *build* de *Linux*, da criação de cada servidor, passando por todos os processos descritos no capítulo anterior até a entrega deste, 7 horas eram gastas no total.

### 3.2.2 Pontos de dificuldade no processo de *build*

O principal ponto de dificuldade informado pelo time é a questão do tempo demandado para as atividades. Gastava-se muito tempo com tarefas repetitivas, sendo necessário fazer o mesmo processo várias vezes para vários servidores. Com a falta de mão de obra, o serviço também acabava acumulando.

Conforme figura 3, os usuários precisavam se *logar* em diversos sistemas e ambientes para realizar suas atividades.

**Figura 3 - Diagrama processual antes da automação.**




**Fonte: Autor**

Outro ponto importante é o erro humano. Como todos os processos eram manuais, erros de digitação ou de ordem de execução eram constantes, fazendo com que o processo se tornasse mais longo e com chances de falha.

Por fim, havia também a falta de centralização. Cada profissional do time tinha um processo salvo em sua máquina diferente do outro, além de versões de instaladores dos aplicativos que também se diferenciavam entre si.

# 4 Solução implementada

A solução que foi aplicada é a automatização dos processos manuais abordados no subcapítulo 3.2, através de *playbooks* do *Ansible*, utilizando o *Jenkins* como *front-end* para que os usuários do time de *Linux* possam preencher um formulário com as variáveis desejadas e executar da maneira mais simples possível.

Todo o processo foi traduzido em *playbooks*, estes foram salvos no servidor onde o *Jenkins* está instalado, desta forma, o próprio *Jenkins* consegue executá-los no servidor que estiver passando pelo processo de *build*.

Devido à falta de mão de obra, o serviço acabava acumulando. Com isso, a solução foi solicitada ao nosso time (time de automação).

## 4.1 Fases da implementação da solução

Primeiramente, foram gastos 3 meses de estudos das ferramentas Ansible e Jenkins. Após isso, iniciou-se o projeto, que foi dividido em quatro fases listadas a seguir:

1. Criação dos *playbooks* - Criação de *playbooks* baseados nas instruções passadas pelo time de *Linux*;
2. Testes em laboratório - Testar os *playbooks* criados na etapa anterior em máquinas virtuais que simulam os servidores da infraestrutura o cliente, a fim de passar para a fase 3 com o mínimo de erros possível.
3. Testes em servidores reais - Criar interface no *Jenkins* para facilitar o teste dos *playbooks*, e solicitar ao time de *Linux* que faça os *builds* necessários para testarmos a solução no ambiente do cliente.
4. Ajustes no *playbooks* - Fazer pequenos ajustes relativos aos erros provenientes da fase 3.

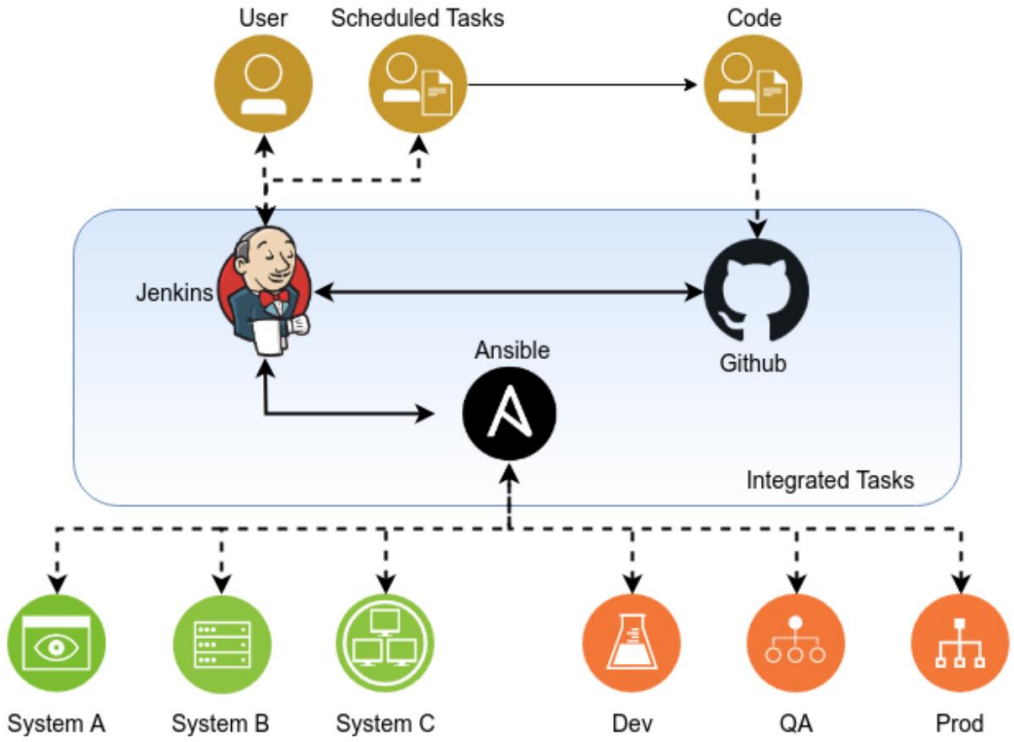
## 4.2 Estrutura final da solução

Para comportar a solução, um servidor foi solicitado ao cliente. Nele, foram instalados o *Jenkins* e o *Ansible*, e também armazenados os *playbooks* e os instaladores necessários para a execução do *build*.

Também foi criado uma rotina de versionamento de toda a solução através do *Git*, sendo armazenado no *GitHub* da *IBM*.

Como descrito na figura 4, com a solução, o usuário apenas precisa se autenticar no *Jenkins* e executa os *builds* por lá, eliminando a necessidade de se *logar* em vários ambientes e servidores.

**Figura 4 - Diagrama processual após a automação.**




**Fonte: Autor**

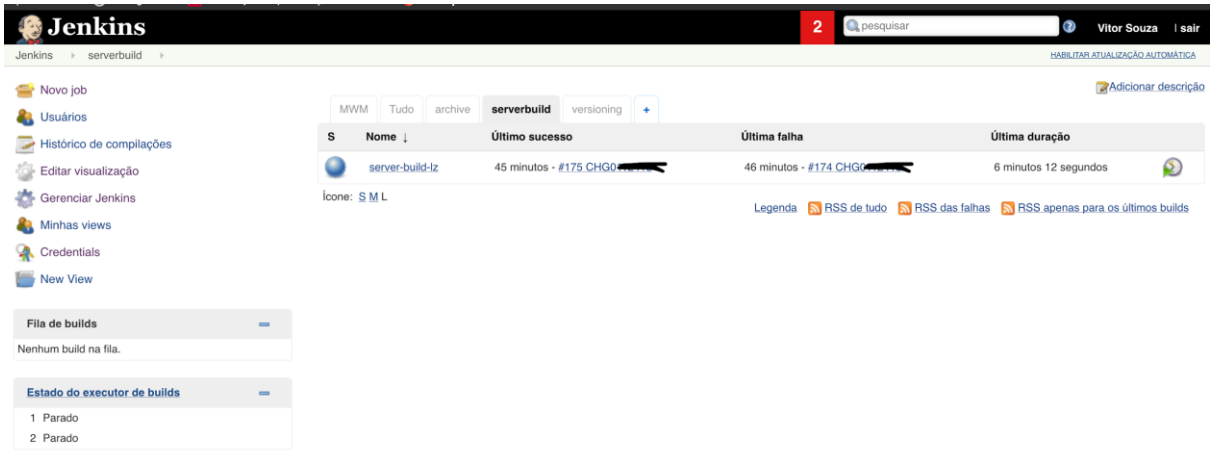
## 5 Resultados da implementação da automação

### 5.1 Novos processos de *build*

Antes de iniciar a execução do *build*, o analista precisa criar um usuário com permissão equivalente a *root*, com uma senha pré-definida. Ele deverá utilizar esse usuário e senha ao executar o *playbook* via *Jenkins*.

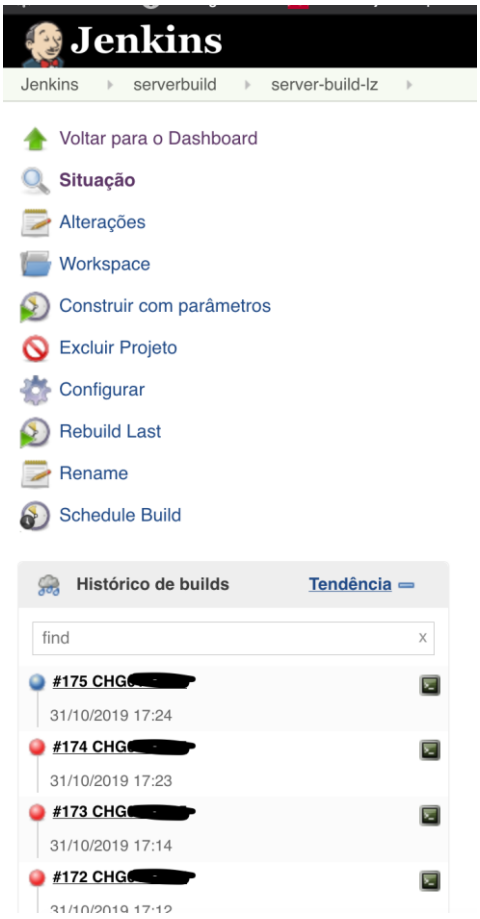
Conforme figuras 5 e 6, o usuário precisa clicar no *link server-build-lz* e em seguida selecionar a opção ‘construir com parâmetros’. Nesta última tela, também é possível a visualização de todos os *jobs* executados, com todo o histórico do que foi feito em cada um.

**Figura 5 - Tela inicial do *Jenkins*.**




**Fonte: Autor**

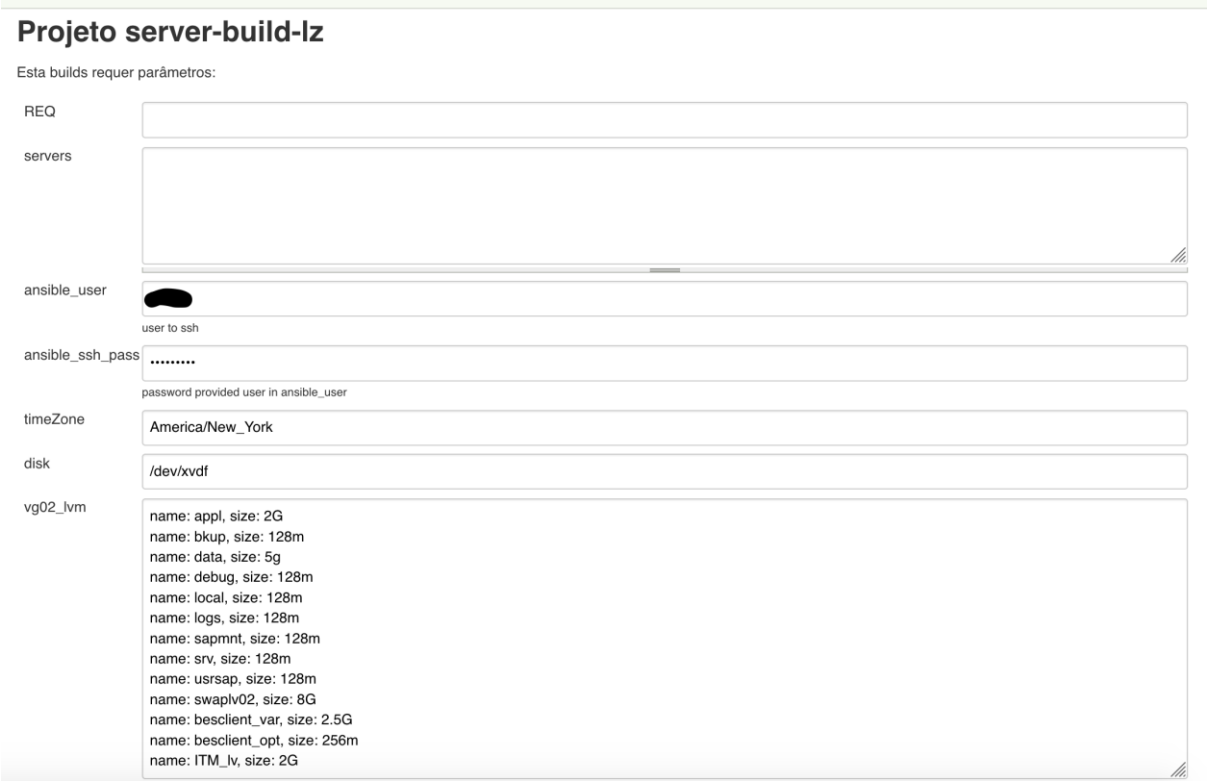
**Figura 6 - *Job* de *build* de servidor.**




**Fonte: Autor**

Após isso, ele precisa preencher o formulário de execução. Algumas informações já vêm pré-definidas, como demonstrado nas figuras 7 e 8, porém o analista pode alterar conforme necessário.

**Figura 7 - Primeira parte do formulário do *Jenkins*.**

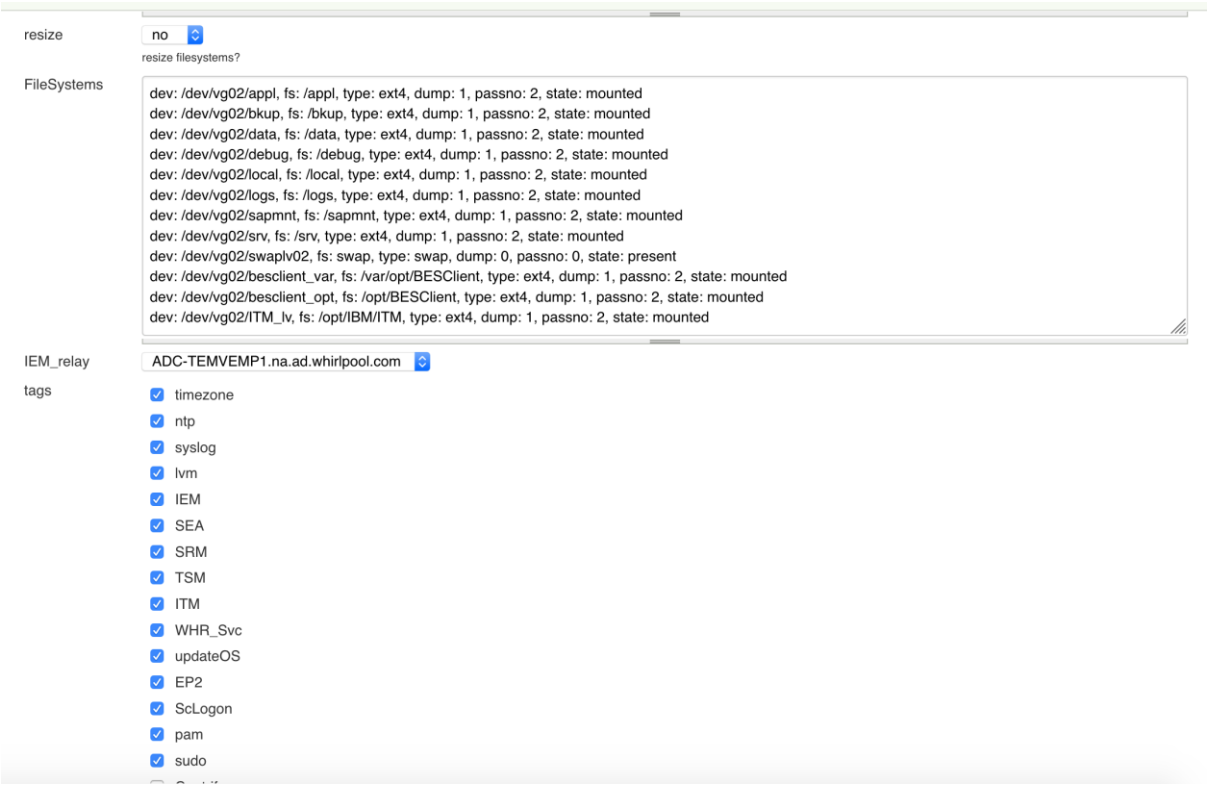



**Fonte: Autor**

Os dados solicitados na figura 7 devem ser preenchidos da seguinte forma:

- *REQ* - Número da requisição associada ao novo servidor.
- *Servers* - *Hostname* ou *IP* dos servidores que passarão por *build.*
- *Ansible_user* - Usuário que executará o build. Precisa de perfil *root*.
- *Ansible_ssh_pass* - Senha do usuário acima.
- *TimeZone* - Qual fuso horário será utilizado.
- *Disk* - O nome do disco em que as aplicações serão instaladas.
- *Vg02_lvm* - Nomes e tamanhos dos discos lógicos a serem criados.

**Figura 8 - Segunda parte do formulário do *Jenkins***



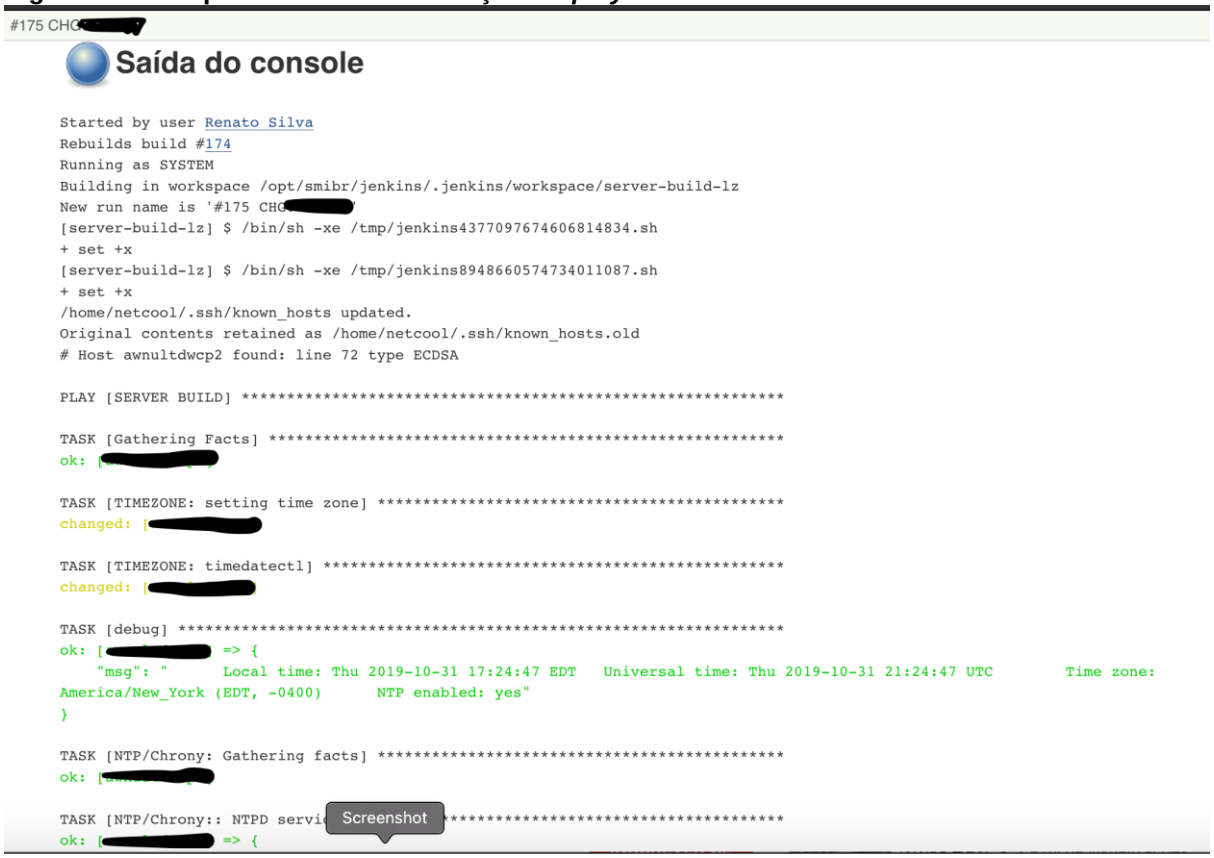

**Fonte: Autor**

A segunda parte do formulário descrito na figura 8, deve ser preenchido da seguinte forma:

- *Resize* - Definir se os sistemas de arquivo devem ou não serem redimensionados
- *FileSystems* - Definir as configurações de sistemas de arquivo.
- *IEM_relay* - Define qual o servidor destino da conexão do *IEM* (Uma aplicação da *IBM*)
- *Tags* - Escolher quais etapas do *playbook* serão executadas. Por padrão, todas elas vêm selecionadas.

Feito isso, o analista somente precisa executar o *playbook* clicando no botão ‘construir’. Ele recebe um *output* em tempo real do que está sendo executado no servidor, conforme figura 9.

**Figura 9 - Exemplo de saída de execução do *playbook*.**




**Fonte: Autor**

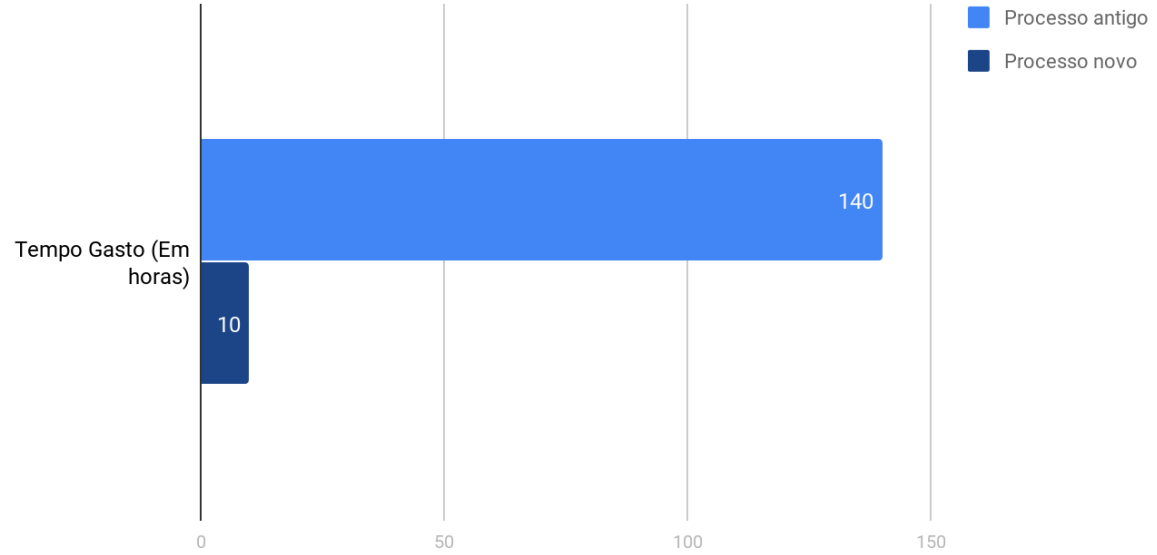
## 5.2 Tempo gasto com o *build* no novo processo

Segundo apurado pelo time de *build* de *Linux*, começando pela criação do servidor, execução de pré *tasks*, preenchimento de formulários do *Jenkins* e execução dos *playbooks*, ao total gasta-se em média 30 minutos por servidor.

Lembrando que a parte de executar os *playbooks* pode ser feita para mais de um servidor por vez, sem limite para número de máquinas.

A média de novos servidores *Linux* no último ano foi de 20 por mês. Com o processo antigo, (de 7 horas para cada servidor) o tempo total gasto seria de 140 horas, ou 17,5 dias úteis de trabalho. Com o processo novo, (30 minutos para cada servidor) o tempo é de apenas 10 horas (Pouco mais de um dia útil). Sendo assim, o tempo total economizado foi de 130 horas, ou 16,25 dias úteis de trabalho, conforme demonstrado na figura 10.

**Figura 10 - Tempo mensal gasto em builds de servidores Linux**




**Fonte: Autor**

## 5.3 Antes e depois da implementação da automação

Na tabela 1 são mostradas as alterações processuais feitas pela automação implementada.

**Tabela 1 - Comparação entre antes e depois da implementação da automação.**

| Antes | Depois |
|---|---|
| Cada analista possuía um procedimento diferente em sua máquina de como realizar o *build.* | Processo definido nos *playbooks* do *Ansible*. |
| Cada analista possuía uma versão diferente dos instaladores dos aplicativos necessários para o *build*. | Instaladores ficam no servidor central da automação, com todos os analistas utilizando os mesmos pacotes. |
| Analista precisava *logar* em cada servidor e realizar todas as atividades necessárias. | Analista *loga* apenas no *Jenkins*, que chama os *playbooks* de *build*, que realizam a instalação em vários servidores de uma vez. |
| Tempo médio de 7 horas por servidor. | Tempo médio de 30 minutos por servidor. |
| Possibilidade de erro humano ao digitar algo errado. | Os processos definidos nos *playbooks* (testados diversas vezes antes de entrar em produção) evitam erros humanos. |

**Fonte: Autor**

## 6 Considerações finais

A utilização de ferramentas de automação (como o próprio *Ansible*) vem se tornando cada vez mais comum no universo da tecnologia da informação, nas mais diversas empresas e comunidades. Essas implementações têm conseguido fazer com que seja economizado tempo e dinheiro, encurtando os processos e diminuindo o erro humano.

Este relatório visou mostrar com detalhes e números o quanto se pode ganhar ao automatizar os processos com uma dessas ferramentas, ao descrever como foi o processo de implementação em um cliente da *IBM*.

Com isso, pôde-se verificar que muitas horas de trabalho foram economizadas, além de erros diminuídos e processos centralizados. Também foi exposto que, sem gastar um centavo (através de ferramentas *open source)*, é possível ter diversos ganhos e melhorias, necessitando apenas investir no próprio conhecimento daqueles que vão implementar a solução.

Apesar dos resultados terem sido bastante interessantes, a melhoria não pode parar. Para essa solução, foi adotado a postura de *continuous improvement,* que é quando se busca melhoria constante, através de interações com o time que executará os *builds* de servidores. Com o *feedback* dos mesmos, planos de melhoria serão traçados e executados para que a automação se torne cada vez mais eficaz.

Também está no radar a implementação de automações para servidores *Windows*, ainda em fase de testes.

## Referências bibliográficas


*CISCO*. ***Configuring secure shell on routers and switches running Cisco IOS***. [s. l.], [s. d.]. Disponível em: <https://www.cisco.com/c/pt_br/support/docs/security-vpn/secure-shell-ssh/4145-ssh.pdf>. Acesso em: 5 nov. 2019.

FUGGETTA, Alfonso. *Open source software - an evaluation.* ***Journal of Systems and Software***, Amsterdam, v. 66, n. 1, p. 77-90, 15 abr. 2003.

*JENKINS*. ***Jenkins user documentation***. [*s. l.*], [s. d.]. Disponível em: <https://jenkins.io/doc/>. Acesso em: 5 nov. 2019.

MAES, Pattie. *Intelligent software: easing the burdens that computers put on people*. **IEEE Expert** 11(6): 62-63, 1996.

MANZUETO, Mauricio Santos. **Automação de processos:** a influência dos softwares de automação de processos nas rotinas organizacionais. [*s. l.*], 2016. Disponível em: <https://www.maxwell.vrac.puc-rio.br/28434/28434.PDF>. Acesso em: 5 nov. 2019.

*RED HAT*. ***Intro to playbooks***. [*s. l.*], [s. d.]. Disponível em: <https://docs.ansible.com/ansible/latest/user_guide/playbooks_intro.html#about-playbooks>. Acesso em: 5 nov. 2019.

*RED HAT*. **O que é a automação de TI?** [s. l.], [s. d.]. Disponível em: <https://www.redhat.com/pt-br/topics/automation/whats-it-automation>. Acesso em: 5 nov. 2019.

*RED HAT*. ***Red Hat Ansible automation***. [*s. l.*], [s. d.]. Disponível em: <https://www.redhat.com/pt-br/technologies/management/ansible>. Acesso em: 5 nov. 2019.

*RED HAT*. ***Use case configuration management***. [*s. l.*], [s. d.]. Disponível em: <https://www.ansible.com/use-cases/configuration-management>. Acesso em: 5 nov. 2019.

*RED HAT*. ***Working with playbooks***. [*s. l.*], [s. d.]. Disponível em: <https://docs.ansible.com/ansible/latest/user_guide/playbooks.html?extIdCarryOver=true&sc_cid=701f2000001OH7EAAW>. Acesso em: 5 nov. 2019.

*RED HAT*. ***YAML syntax***. [*s. l.*], [s. d.]. Disponível em: <https://docs.ansible.com/ansible/latest/reference_appendices/YAMLSyntax.html>. Acesso em: 5 nov. 2019.

REIS, Christian Robottom. **Caracterização de um processo de *software* para projetos de *software* livre**. Orientador: Profa. Dra. Renata Pontin de Mattos Fortes. 2003. 247 p. Dissertação (Mestrado em ciências da computação) - Universidade de São Paulo, São Carlos, 2003. DOI 10.11606/D.55.2003.tde-12112014-100100. Disponível em: <https://teses.usp.br/teses/disponiveis/55/55134/tde-12112014-100100/publico/ChristianRobottomReis.pdf>. Acesso em: 5 nov. 2019.