CURSO DE SISTEMAS DE INFORMAÇÃO

PROPOSTA DE UM MODELO DE IMPLANTAÇÃO PARA APLICAÇÃO EXTERNA UTILIZANDO CONCEITOS DEVOPS

MATHEUS JOSÉ ALVES SILVA SANTOS

Palmas

2018

CURSO DE SISTEMAS DE INFORMAÇÃO

PROPOSTA DE UM MODELO DE IMPLANTAÇÃO PARA APLICAÇÃO EXTERNA UTILIZANDO CONCEITOS DEVOPS

MATHEUS JOSÉ ALVES SILVA SANTOS

Projeto apresentado como requisito para aprovação na disciplina de Estágio Supervisionado do Curso de Sistemas de Informações da Universidade Estadual do Tocantins- UNITINS, sob a orientação do professor Me. Douglas Chagas da Silva.

Palmas

2018

1. INFORMAÇÕES DO ACADÊMICO:

Nome: Matheus José Alves Silva Santos Matrícula:

Período 7

E-mail: matheusetf@gmail.com Telefone: (63) 99285-7729

2. INFORMAÇÕES DO ESTÁGIO:

Professor Orientador: Douglas Chagas da Silva

Início do Estágio: Término do Estágio: 16/06/2018

Total de horas semanais dedicadas ao estágio supervisionado: 20 horas

Área de realização do estágio: Infraestrutura Ágil

Data: ____/____/____

3. PARECER DO PROFESSOR ORIENTADOR:

Aprovado ( ) Reprovado ( ) Nota: _____

OBSERVAÇÕES:

DATA: ____/____/____

_____

Professor Orientador

4. PARECER DO COORDENADOR DE ESTÁGIO:

OBSERVAÇÕES:

DATA: ____/____/____

_____

Coord. de Estágio Supervisionado

Palmas

2018

Este trabalho é dedicado à minha família, pelo apoio incondicional.

Agradecimentos

À Deus, pela graça de permitir o meu crescimento no campo de estudo em que me sinto feliz e realizado.

Aos meus pais e meus irmãos que sempre apostaram em minha capacidade de crescimento e superar as barreiras que a vida impõe.

À minha esposa que tem sido compreensiva nos momentos de dificuldades, e companheira nos desafios que surgem.

Aos meus amigos e colegas de universidade, que sempre contribuíram para o desenvolvimento do meu conhecimento e pelas parcerias nos projetos e estudos.

"Não só isso, mas também nos gloriamos nas tribulações, porque sabemos que a tribulação produz perseverança; a perseverança, um caráter aprovado; e o caráter aprovado, esperança. E a esperança não nos decepciona, porque Deus derramou seu amor em nossos corações, por meio do Espírito Santo que Ele nos concedeu."

(Bíblia Sagrada, Romanos 5:3-5)

Resumo

Este projeto visa apresentar um estudo sobre as diversas ferramentas de automação da entrega de software, visando estabelecer uma proposta de cenário que viabilize esse processo dentro de uma cultura DevOps. O cenário deve funcionar como um provedor de serviços sobre plataforma ágil.

Palavras-chaves: Automação, DevOps, Modelo Ágil.

Lista de ilustrações

Lista de abreviaturas e siglas

DevOps - Development and Operational (Desenvolvimento e Operacional).

AWS - Amazon Web Services

SSH - Secure Shell

NETCONF - Network Configuration

Sumário

1 Introdução

As mudanças econômicas refletem demandas de mercado. Isso têm impacto fundamental na forma como os recursos são disponibilizados e os serviços são exigidos. Nesse sentido, a tecnologia da informação, e mais especificamente nesse estudo, a entrega de um produto ou serviço deve dar um suporte a altura das cobranças de mercado.

O cenário atual que envolve os processos de desenvolvimento e infraestrutura estão ficando cada vez mais defasados. Tradicionalmente o processo de *deploy*[1] e gestão de entrega de soluções, tende a tornar a rotina de produção e disponibilização de serviços demasiadamente morosa. Em termos atuais o usuário final e o próprio mercado demanda agilidade e rápida resposta.

A concepção de um modelo de entrega contínua que permita ganho na produtividade e fácil detecção de erros no processo de *deploy*, tem sido cada vez mais requisitado no meio da gestão da tecnologia de informação aliado a uma cultura *DevOps*[2]. Tomando isso como base, é importante entendermos que os prazos de entrega estão cada vez mais apertados, o que influi no aumento da carga de trabalho, refletindo diretamente no desempenho do profissional, tanto de desenvolvimento quanto de infraestrutura.

O presente projeto tem como foco a formulação de um modelo que permita essa entrega contínua e automatizada no que diz respeito o *deploy* de aplicações em tempo minimamente reduzido. Passa-se antes pelo estudo das diversas ferramentas disponíveis no mercado de software, a fim de levantar-se as possibilidades de aplicação nesse modelo. Sendo assim serão apresentadas as ferramentas eleitas como mais viáveis para o projeto.

É vital então percebermos a real vantagem no uso do modelo baseado na visão DevOps, a redução na carga de trabalho e a comunicação massiva entre os times de infraestrutura e desenvolvimento dentro de um mesmo projeto, e a entrega ágil do produto trabalhado.

A estrutura desse trabalho está organizado da seguinte forma: primeiramente são traçados os objetivos. Na seção 2 será apresentado o referencial teórico utilizados para embasar o tema e a ideia proposta de acordo com todas as ferramentas estudadas a fim de utilização posterior em um cenário fictício; na seção 3 será apresentado os métodos utilizados para compor o estudo, assim como as ferramentas utilizadas durante todo o processo. Na seção 4, são apresentados os resultados referente ao projeto. Por fim, menciona-se as referências utilizados no referido estudo.

[1] Publicação de um determinado software ou serviço para uso.
[2] Development and Operational (Desenvolvimento e Operacional).

1.1 Objetivos

1.1.1 Objetivo Geral

Apresentar um modelo de arquitetura de aplicação externa para entrega contínua baseada na visão DevOps.

1.1.2 Objetivos Específicos

- Diferenciar o modelo tradicional de deploys e a visão DevOps;
- Pesquisar as ferramentas disponíveis mais relevantes que viabilizam a entrega contínua para aplicação de terceiros;
- Definir os benefícios do uso da visão DevOps numa aplicação;
- Apresentar o ganho real na produtividade e tempo, no uso da entrega contínua.

2 Referencial Teórico

O conceito de DevOps pode ser entendido como o nivelamento entre as equipes de desenvolvimento e operações no que tange suas interações, mantendo suas funções específicas, porém alinhando suas demandas referentes às responsabilidades e processos, visando a disponibilização de um produto ou funcionalidade de forma rápida e confiável.

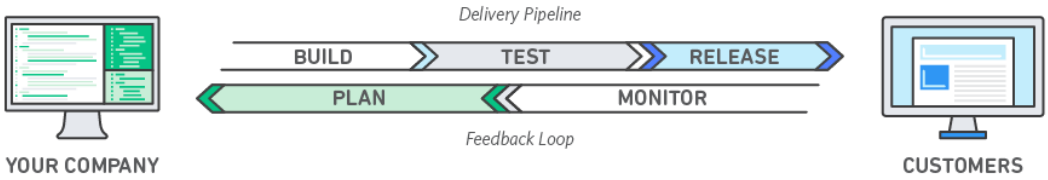

Figura 1 – Ciclo DevOps
Fonte: Amazon Web Services, Inc., 2018

As implementações nesse tipo de ambiente fazem uso de ferramentas de automação com o intuito de dinamizar cada vez mais a infraestrutura e torná-la mais programável, de forma que reflita na melhoria contínua da comunicação e integração entre desenvolvedores e administradores de infra, transformando o cenário tradicional de isolamento entre essas duas equipes em um ambiente participativo e colaborativo.

O objetivo do DevOps é gerar em toda a equipe envolvida na produção de um software, uma cultura que vise o aumento do fluxo de trabalho (maior frequência de deploys) e em paralelo a isso dar mais robustez no desenvolvimento da aplicação. Esse conceito representa muito mais do que simplesmente o uso de ferramentas de automação, é importante observarmos que trata-se de uma quebra de paradigmas e uma mudança na cultura no negócio com uma nova forma de produção.[1]

A agilidade no processo de deploys citado anteriormente aluz à uma necessidade de amparo para que essa entrega rápida de fato aconteça, e mais do que isso, a demanda por parte dos clientes ou usuários de determinada aplicação, requer cada vez mais velocidade no uso de determinada funcionalidade. Analogamente à revolução industrial do século XX, a mudança na forma de produção de produtos naturalmente está sendo absorvida pela tecnologia, assim, a concepção desse produto deve acompanhar a demanda externa. Nesse sentido, a adaptabilidade e mudança da arquitetura funcional no desenvolvimento de serviços tecnológicos são necessidades relevantes, a cultura DevOps é uma mudança importante nesse ponto.

Segundo uma pesquisa realizada pelo Gartner Group sobre DevOps, em 2015, somente 29% das organizações pesquisadas tem o modelo atuante em produção. É evidenciado ainda que apenas 42% desses, tem a atuação do DevOps em aplicações móveis.

De acordo com o mesmo grupo o DevOps evoluiria de uma estratégia de nicho para uma estratégia comum sendo empregada por 25% das organizações do Global 2000[1].

O uso da cultura DevOps deve ser absorvida na necessidade de versionamento contínuo de determinada aplicação, ou seja, a frequente execução de deploys. Nisso, é importante observarmos que a rápida proliferação de software requer atualizações operando na mesma medida ágil e demanda pela competitividade de mercado, visto que a velocidade no atendimento da expectativa dos clientes diferencia a empresa em relação às demais atuantes no mercado.

A principal vantagem no uso de DevOps é a melhora evidente nos processos e automatização das tarefas, otimizando o tempo e reduzindo os ciclos de desenvolvimento. Por se tratar de uma interação entre as equipes de desenvolvedores e operacionais, dizemos que é um sistema bimodal de trabalho.[2]

O monitoramento de métricas e registro de logs é um aspecto relevante. Leva-se em conta que os serviços devem estar disponíveis 24 horas por dias e durante os 7 dias da semana, acarretando em uma massiva análise de dados e logs gerados pelo sistema, portanto, a rotina na observância desses elementos deve ser constante. É possível, inclusive, a criação de alertas que apontem situações e permitam a gerência proativa dos serviços.

Outro benefício fundamental do DevOps é o aumento na comunicação e colaboração que envolve todos os personagens da empresa. É importantíssimo a definição de normas que permitam um maior compartilhamento de informações, ou que permita a proliferação da comunicação, seja qual for o método ou tecnologia, desde que agregue valor. Além disso, a diminuição de ruídos na comunicação e conflitos entre as equipes melhora o ambiente e tende a produzir efeitos positivos ao fim do processo.

Podemos ainda elencar ganhos em maior estabilidade e melhor desempenho, e tão importante quanto, a redução considerável de custos de trabalho, visto que a diminuição de tempo de produção e menor esforço afeta diretamente o custo estimado em um projeto.

2.1 Infraestrutura como Código

Falar em infraestrutura com código é exatamente absorver o entendimento do tratar a estrutura de TI como um software, programável. Como isso é possível o uso de práticas que envolvam o controle por versionamento, testes automatizados, entrega contínua, entre diversos outros recursos por meio de scripts específicos.

O uso de práticas anteriores com métodos de gerenciamento de infraestutura foram válidos e deram uma base poderosa para a concepção de novas tecnologias. Atualmente,

[1] Forbes Global 2000 é uma classificação anual das 2.000 empresas públicas do mundo pela revista Forbes. O ranking é baseado em quatro critérios: vendas, lucro, ativos e valor de mercado. A lista é publicada desde 2003. Fonte: Forbes

há uma constante demanda de software cada vez mais dinâmicos e um alto índice de deploys, inclusive simultâneos, levando à necessidade de concepção de uma infraestrutura que acompanhe o ritmo de complexidade desses novos sistemas.

Segundo a Hewlett Packard[2], em seu site oficial, cita que a infraestrutura como código elimina a necessidade de criar diversos ambientes de produção de forma manual, isolada e separadamente, e/ou atuailzações de hardware e sofware. Toda essa dinâmica pode ser feita através de scripts contidos no mesmo conjunto de códigos, trazendo velocidade, economia e otimização de tempo. Nesse contexto, a infraestrutura como código traça uma linha tênue entre o código que executa a aplicação e o código que configura um ambiente, tornando um ambiente característico do DevOps.

2.2 Principais Ferramentas e Recursos Tecnológicos Disponíveis

Apesar do conceito DevOps ser recente, a gama de ferramentas que contribuem para implantação dessa cultura já se mostram bastante diversificadas. Dentre as mais comuns podemos apresentar:

2.2.1 Atlas[3]

É uma ferramenta disponibilizada pela Hashicorp, que tem a função de unificar projetos open source para o manejo de aplicações finalizadas no desenvolvimento para a produção em qualquer que seja a infraestrutura. As etapas do Atlas seguem cinco passos (vide figura 2), isso independe da tecnologia utilizada, sejam máquinas virtuais ou contêineres, as etapas se mantêm as mesmas.

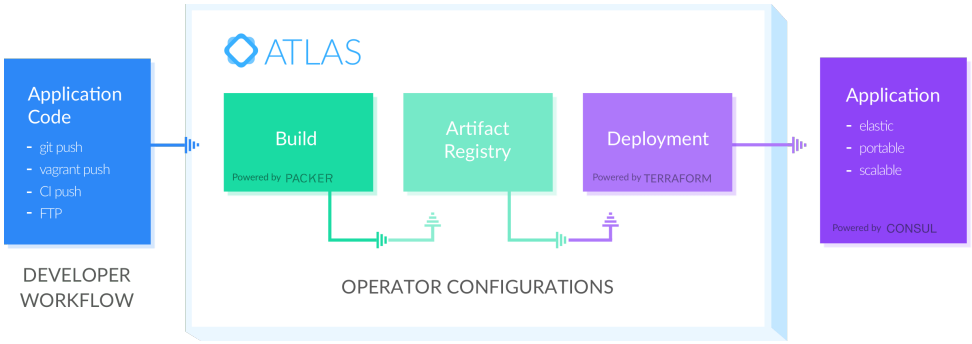

Figura 2 – Etapas do Atlas
Fonte: Hashicorp[4]

O Atlas não é um software de caixa preta, ou seja, é possível acesso a serviços que contribuem para tanto, como Vagrant (gerencia ambientes de desenvolvimento), Packer

[2] Disponível em: https://www.hpe.com/br/pt
[3] Disponível em: https://www.hashicorp.com/blog/atlas-announcement

(construção de artefatos), Terraform (implantação de Infraestrutura) e Consul (monitora os serviços em tempo real).

2.2.2 Chef[5]

É um framework destinado à automatização para sistemas e infraestrutura em nuvem. O Chef constrói, entrega e administra fazendo uso de scripts replicáveis.

O Chef tira a carga dos administradores de sistemas que focam no gerenciamento projetado para servidores autônomos, ele permite executar cetenas de instâncias de servidor em um tempo imensamente maior se comparado ao uso comum em deploys.

Para gerenciar esse tipo de configuração, o Chef, transforma a infraestrutura em código, deixando o processo mais flexível, legível pelos analistas e testável, possibilitando assim a gerência de recursos tanto localmente quanto na nuvem

Em sua topologia o Chef tem três principais componentes: o Servidor Chef, as Estações e o Nós.

O maior atrativo dessa ferramenta é o uso de "cookbooks"ou receitas, são ditas configurações plugáveis, que envolvem todas as instalações e parâmetros necessários para atender determinada aplicação no servidor ou máquina. Assim como as receitas convencionais definem uma estrutura sequencial que deve ser seguida a fim de tornar o produto final reflexo de uma ideia original, o Chef mantém um conceito semelhante a isso, onde define-se um estado desejado do sistema, desenvolvendo um código de configuração, então o ele processa esse código, une ao processo os dados do nó em questão e garante que o estado concebido seja correspondente ao estado do sistema.

O Chef pode ser executado em várias plataformas, como Windows, distribuições Linux, FreeBSD, Solaris, AIX, Cisco IO e Nexus. E ainda suporta plataformas em nuvem,como Amazon Web Services (AWS), Google Cloud Platform, OpenStack, IBM Bluemix, HPE Cloud, Microsoft Azure e VMware vRealize.

2.2.3 Docker[6]

O Docker é uma plataforma de automação que implanta as aplicações em espaços isolados chamados de *contêineres*, possibilitando as executarem as aplicações de forma mais ágil. O objetvo é criar múltiplos ambientes dentro de um mesmo servidor, acessíveis externamente.

Segundo Matthias e Kane, no livro Primeiros Passos com Docker, essa ferramenta a arquitetura do software de modo a tornar o armazenamento ainda mais robusto. A essência

[5] Disponível em https://www.chef.io
[6] Disponível em https://www.docker.com/

no Docker é acomodar os serviços ou aplicações em contêineres atômicos ou descartáveis. Em todo o processo de desenvolvimento um contêiner pode ser excluído e ser restaurado se assim surgir a necessidade, isso torna a dinâmica de entrega e teste extremamente flexível.[3]

O Docker é uma plataforma Open Source, que não pode ser confundido com um ambiente tradicional de virtualização. No Docker fazemos uso de recurso isolados que utilizam bibliotecas do kernel em comum, isso porque é presente nessa ferramenta o Linux Containers (LXC).

O mesmo permite empacotar a aplicação ou um ambiente em um contêiner e movê-lo para qualquer outro host que possua o Docker instalado, tornando assim uma ferramenta portável. A grande vantagem disso é que não há necessidade de reconfigurar todo o ambiente novamente, visto que todo ele é movido, reduzindo acentuadamente o tempo de deploy de uma infraestrutura.

Essa plataforma não pode ser confundida com um ambiente virtualizado, visto que em cenários que utilizam máquinas virtuais há a presença de uma camada intermediária de sistema operacional entre o host e as aplicações, no Docker essa é desnecessária pois ele não utiliza kernel, tornando independente quanto a nível de disco, memória e processamento.[4]

A infraestrutura no Docker também é replicável, é possível criar imagens predefinidas e disponibilizá-las em ambientes de desenvolvimento, teste, homologação e produção para aplicações.[5]

2.2.3.1 Vantagens:

Podemos elencar alguns ganhos consideráveis na utilização do Docker[6]:

- Empacotamento de software otimizando o uso das habilidades dos desenvolvedores;
- Empacotamento de aplicação de software com todos os arquivos e dependências necessárias para determinada aplicação.
- Utilização de artefatos empacotados que possibilitem a passagem pelo teste e produção sem necessidade de recompilação.
- Uso de softwares sem onerar recursos demasiados, visto que o contêiner é apenas um processo que se comunica diretamente com kernel do Linux.

2.2.4 Puppet[8]

É uma ferramenta de código livre para gestão de configurações. A ideia central do Puppet é a administração de diversas máquinas físicas ou virtuais, onde a configuração é

[8] Disponível em https://www.puppet.com/

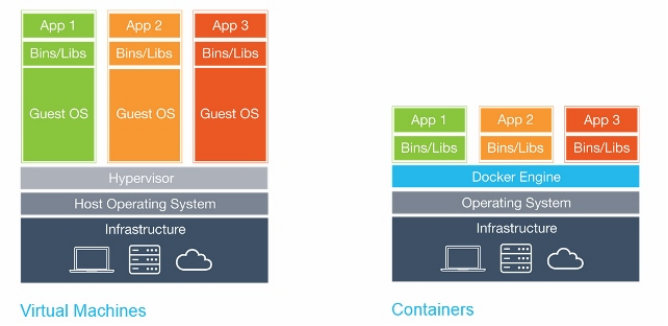

Figura 3 – Comparativo Docker x Máquinas Virtuais
Fonte: Bright Computing[7]

centralizada em um único nó e então distribuídas por diversos nós na rede, assim, gerenciando configurações, automatizando instalação de pacotes e facilitando o estabelecimento de normas e auditoria.[7]

A ferramenta está disponível em duas versões: Puppet Enterprise[9] (com suporte pago), e a Open Source Puppet[10] (código aberto). É uma ferramenta bem utilizada pela comunidade e por isso existem muitos módulos desenvolvidos, empresas como McAfee e Nasa fazem uso dela.

O Puppet utiliza SSH para a conexão aos hosts, esse é um ponto positivo se olharmos pela ótica que em alguns cenários não é possível a instalação de agentes ou em situações onde o agente consome uma fatia considerável de memória e cpu.

	Ansible	Puppet	Chef
Script Language	YAML	Custom DSL based on Ruby	Ruby
Infrastructure	Controller machine applies configuration on nodes via SSH	Puppet Master synchronizes configuration on Puppet Nodes	Chef Workstations push configuration to Chef Server, from which the Chef Nodes will be updated
Requires specialized software for nodes	No	Yes	Yes
Provides centralized point of control	No. Any computer can be a controller	Yes, via Puppet Master	Yes, via Chef Server
Script Terminology	Playbook / Roles	Manifests / Modules	Recipes / Cookbooks
Task Execution Order	Sequential	Non-Sequential	Sequential

Figura 4 – Comparativo entre Ferramentas de Gerenciamento de Configurações
Fonte: Digital Ocean[11]

[9] Disponível em: https://puppet.com/products/puppet-enterprise
[10] Disponível em: https://puppet.com/download-open-source-puppet

2.2.5 Ansible[12]

O Ansible foi criado em 2012, por Michael DeHann, basicamente essa ferramenta gerencia configurações e orquestra tarefas. Nesse sentido ele implementa módulos para nós sobre SSH. Os módulos são distribuídos nos nós temporariamente que realizam a comunicação com a máquina de controle (assim com em ferramentas concorrentes) por meio de um protocolo JSON[13].

O diferencial do Ansible em relação as demais ferramentas que se propõe ao mesmo objetivo, é que ele atua com uma arquitetura cliente-servidor, sem agente, isso que dizer que os nós não são necessários para a instalação dos daemons para comunicação com a máquina controle. Isso resulta em diminuição de carga na rede.

Os objetivos mais notáveis do Ansible é tornar a experiência do usuário muito mais simples e fácil, e investir massivamente na segurança e confiabilidade utilizando o OpenSSH como condutor de dados. É uma linguagem construída temo como parâmetro a auditabilidade, ou seja, possibilitar ao analista o acompanhamento de todas as etapas, rotinas e dados circulados dentro de uma arquitetura definida.[9]

Nessa ferramenta ainda encontramos o uso de um arquivo de inventário, denominado "hosts", que definem quais nós serão gerenciados, que é simplesmente um arquivo de texto que lista os nós individualmente ou agrupados ou até um arquivo executável que constrói um inventário de hosts. É uma opção de alta confiabilidade e segurança, e isso se fundamenta pelo uso do Secure Shell. Possui ainda fácil usabilidade, no entanto, não deixa a desejar em qualquer aspecto se comparado a soluções concorrentes.

Na forma tradicional de trabalho o Ansible faz o upload do código que deve ser executado nas estações clientes, é então executado, retorna o resultado da execução e após isso é removido dos clientes.

Quando usamos a ótica DevOps esse tipo de fluxo é modificado, fazendo uso do protocolo NETCONF[14] (RFC 6241), onde é possivel o envio de comandos aos componentes e receber o retorno da aplicação.

2.2.5.1 Componentes:

O Ansible é estruturado pela composição dos seguintes elementos:

- Playbooks[15]: arquivos de configuração, implementação e linguagem de orquestração do Ansible.

[12] Disponível em https://www.ansible.com
[13] Disponível em: http://jsonapi.org/
[14] Disponível em: https://tools.ietf.org/html/rfc6241
[15] Disponível em https://docs.ansible.com/ansible/playbooks.html

- Agentless[16]: É descartado o uso de agente nos servidores a serem monitorados. Isso deve-se à utilização de OpenSSH para definir o estado atual do ambiente, adaptando-se se acaso estiver em desconformidade com a configuração no playbook.
- Módulos[17]: São as tarefas executadas de fato. Os módulos são ditos como "plugins de tarefas"e por isso são eles que realizam as atividades pertinentes.
- Inventário[18]: Armazena e controla informações sobre os grupos de hosts.

2.2.5.2 Vantagens:

Podemos apontar alguns ganhos consideráveis na utilização do Ansible[10]:

- Não há a necessidade na instalação de agentes nos servidores a serem gerenciados;
- Gerencia paralela e simultaneamente de forma orquestrada;
- Simples configuração e bem estruturado;
- Desenvolvimento em diversas linguagens.

2.2.6 GitHub[19]

É um dos serviços web mais difundidos. Com essa ferramenta é possível hospedar projetos e aplicações, trabalhando com controle de versionamento.

O Github funciona por meio de repositórios, que funcionam como pastas e dentro destas outras pastas que comportam os arquivos de diversas extensões e linguagens. Ainda oferece a possibilidade de contribuir com um determinado para o desenvolvimento de determinado projeto. Além, e permitir o acesso múltiplo por diversos desenvolvedores de um mesma empresa, permitindo que vários programadores trabalhem simultaneamente em uma mesma aplicação ou arquivo.

[16] Disponível em https://dbruno.ansible.com/ansible/
[17] Disponível em https://docs.ansible.com/ansible/modules.html
[18] Disponível em https://dbruno.ansible.com/ansible/
[19] Disponível em https://www.github.com

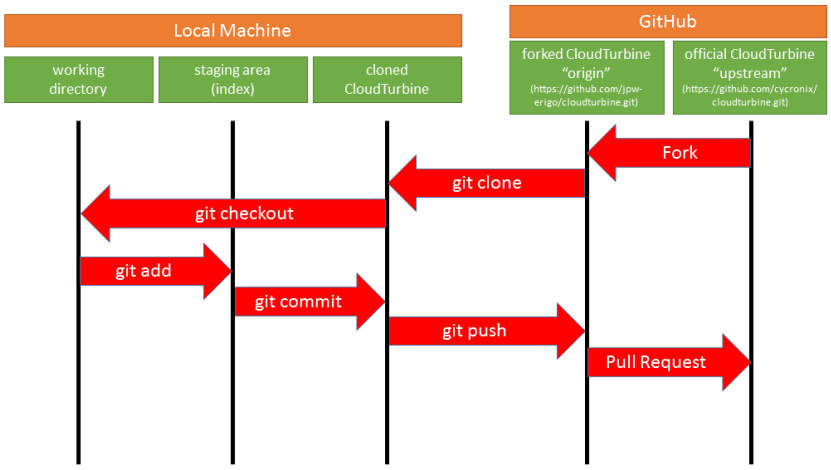

Figura 5 – Fluxo commit e request
Fonte: Cloud Turbine[20]

2.2.7 Jenkins

Essa ferramenta multiplataforma e funciona como um servidor destinado a integração contínua que automatiza a execução de tarefas, possui código aberto e permite ao usuário total liberdade em sua operação.

Basicamente o Jenkins atua em um conceito de integração contínua, onde o objetivo é agilizar tarefas que tradicionalmente demandam]um tempo de excecução mais prolongado como compilação do projeto e execução de testes automatizados. Cada interação é analisada por um build automatizado a fim de detectar possíveis erros de integração, isso permite que testes sejam realizado com o objetivo de reduzir problemas de updates e tornando o software mais coeso.

O Jenkins[21] pode ser integrado ao Git, SVN, CVS, Maven, entre outros. É importante ressaltar que um ponto forte do Jenkins é a difusão entre a comunidade.

Um recurso interessante é o uso de Forks (recurso do GitHub), que permitem a criação de cópias de um determinado projeto e trabalha nesse sem a preocupação de afetar em algum ponto a aplicação original.

É possivel ainda, adicionar testes de desempenho e balanceamento na integração contínua, permitindo a análise de risco e a reduzir eventuais quedas de performance no momento em que um novo recurso é adicionado ou a correção de um erro presente.

2.2.7.1 Vantagens:

- Builds periódicos;
- Testes automatizados;

[21] Disponível em https://www.jenkins.io

- Possibilita análise de código;
- Identificar erros mais cedo;
- Fácil de operar e configurar;
- Comunidade ativa;
- Integra outras ferramentas por meio de plugins nativos.

3 Metodologia

Para o desenvolvimento deste trabalho, com foco em atendimento aos seus objetivos, far-se-á uso de softwares open source (código livre), que possibilitam a utilização gratuita dos mesmos, tendo suporte contínuo da comunidade, além de permitir alterações em sua estrutura, acaso sejam necessárias e relevantes.

No âmbito desse estudo, será construído um cenário hipotético a fim de realização dos testes referentes à deploys em estudo posterior.

Para a eleição das ferramentas escolhidas foi levando em conta, a sua disponibilização de código open source, a flexibilidade no uso e confiabilidade nos resultados, as suas vantagens e desvantagens, a aceitação junto a comunidade científica e a documentação disponível, seja por meio de livros ou artigos publicados ou por contribuições junto a comunidade web.

3.0.1 Ferramentas Utilizadas

Dessa forma, após as devidas análise e tomando como foco a combinação das ferramentas estudadas, optou-se pelas seguintes ferramentas para suportar o estudo:

- Virutalbox: Criação de máquinas virtuais para testes;
- GitHub: Versionamento e repositório;
- Ansible: Gerenciamento de configuração e orquestração da rotina;
- Jenkins: Automatização da execução de tarefas.

4 Resultados

Após Análise das ferramentas estudadas e posteriormente eleitas as que melhor se adequaram para o presente estudo, partiu-se para a elaboração e concepção do modelo proposto para entrega de aplicação externa com base em uma visão DevOps.

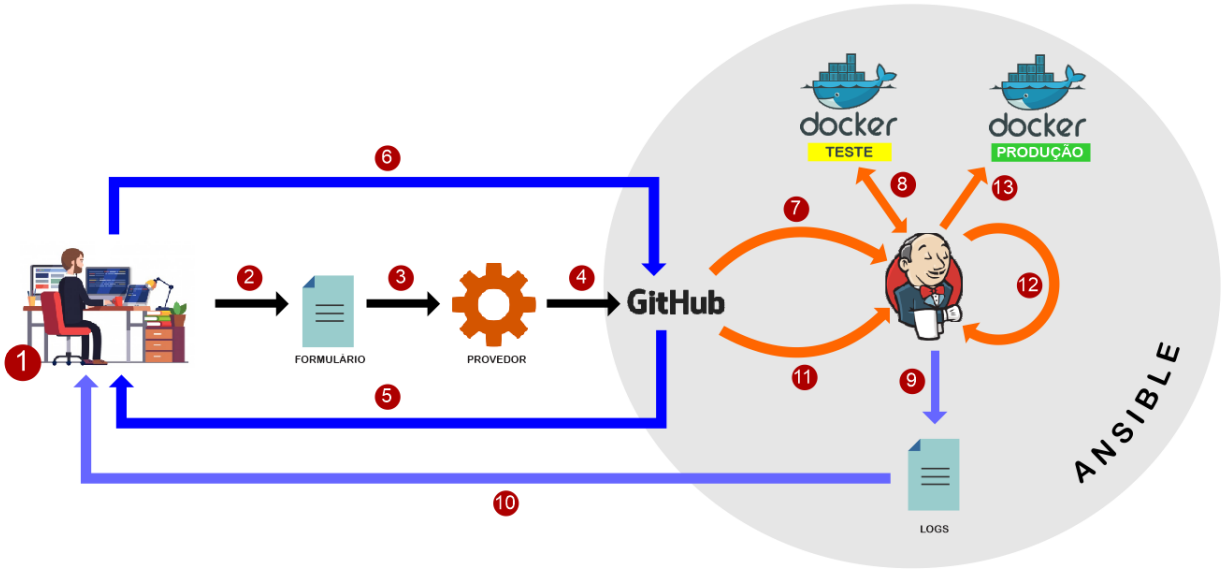

Figura 6

Considerando o modelo acima, temos como sequenciamento os seguintes pontos:

1. No ponto de partida, é pre-requisito a existência de conteúdo codificado em linguagem de programação, que permita o versionamento do projeto.
2. O desenvolvedor faz a requisição de um novo repositório através de um formulário, que contenha as informações pertinentes, tais como, chave ssh, membros do projeto e nome do mesmo;
3. A requisição é recebida pelo provedor que verifica os dados de solicitação, avalia a disponibilidade de repositório;
4. O provedor valida a disponibilidade e cria um novo repositório no GitHub;
5. O link do repositório é retornado ao usuário juntamente com sua chave de autenticação;

6. O desenvolvedor envia os commits e o GitHub gera uma notificação de alteração ao Jenkins;
7. Jenkins recebe a notificação (Git Fetch – Git Merge – Templates Ansible – Dry Run – Commit Check);
8. O Jenkins realiza teste jundo ao contêiner Docker destinado para tanto;
9. Jenkins gera um histórico (Notificação aos Administradores);
10. Administrador analisa os logs de testes;
11. GitHub gera nova notificação ao Jenkins;
12. Jenkins gera novo release e gera nova tag de versão;
13. Estando as alterações aprovadas o Jenkins realiza deploy da aplicação em um contêiner Docker destinado a produção;

5 Conclusão

Este trabalho apresentou conceitos da visão DevOps e analogia com o modelo tradicional de entrega de um software, onde ao decorrer do estudo analisaram-se diversas ferramentas disponíveis no mercado e elegeram-se as que mais se adequavam a necessidade da implementação de um modelo de entrega para aplicação externa com o conceito de deploy automatizado.

O deploy automatizado fornece ao desenvolvedor e profissional de TI um ambiente simplificado para entrega de uma aplicação externa, tornando o gestor de tecnologia um provedor de serviço.

O presente modelo tem como foco contribuir para o aprimoramento no processo de entrega de software. Nesse sentido o cenário atual ainda lida com o processo de deploy baseado no modelo tradicional "cascata", com o estudo realizado nesse projeto é possível otimizar o planejamento do processo da entrega da aplicação para produção utilizando recursos automatizados e gratuitos (open source) e uma visão DevOps. É possível economizar não só tempo como recursos humanos e financeiros que são fundamentais para a conclusão do processo.

Nesse sentido, a utilização de ferramentas open source permite aos profissionais de TI realizarem seu trabalho de forma mais rápida e eficiente. Além disso, diminui o tempo de espera dos corredores por uma hidratação e descanso adequados depois de uma longa corrida. Motivando as duas partes envolvidas no evento. Devido a isso é notória a importância de uma mudança no paradigma de disponibilização de software considerando a rápida demanda de mercado e confiabilidade dos produtos.

Para continuidade deste projeto, objetiva-se a implementação de um cenário funcional utilizando máquinas virtuais, como expõe-se a seguir:

1. Implementar o cenário apresentado neste projeto;
2. Validar o modelo considerando diversas situações;
3. Implementar o modelo visando ambientes críticos e cenários de risco.

Referências

FERRARI, M. A. M. C.; MOREIRA, M. R.; VALDERRAMAS, Z. L. Manual de elaboração e normalização de trabalhos de conclusão de curso – TCC: segundo as normas ABNT e Vancouver. São Paulo: Biblioteca 24horas, 2015.

SATO, Danilo. DEVOPS Na Prática: Entrega de Software Confiável e Automatizada. São Paulo: Casa do Código, 2016.

MATTHIAS, Karl e KANE, Sean. Primeiros Passos com Docker: Usando Contêineres em Produção. São Paulo: Novatec, 2015.

IBM. O que é DEVOPS? 2013. Disponível em: <https://www.ibm.com/developerworks/community/blogs/rationalbrasil/entry/o_que_devops?lang=en>. Acesso em: 12 fev 2018.

GARTNER, Group. Glossário de Ti: DEVOPS. Disponível em: <https://www.gartner.com/it-glossary/devops>. Acesso em: 13 fev 2018.

GARTNER, Group. Newsroom: Gartner Says By 2016, DevOps Will Evolve From a Niche to a Mainstream Strategy Employed by 25 Percent of Global 2000 Organizations. Disponível em: <https://www.gartner.com/newsroom/id/2999017>. Acesso em 13 fev 2018.

GAEA. 4 Motivos para Usar a Cultura DevOps.Disponível em: <https://gaea.com.br/4-motivos-para-usar-a-cultura-devops/#>. Acesso em: 13 fev 2018.

COSTA, Bruna Lima. DevOps, quem usa? como adotar? o que usar?. Disponível em: <https://www.primeinf.com.br/devops-quem-usa-como-adotar-o-que-usar/>. Acesso em 13 fev 2018

HASHIMOTO, Mitchell. Atlas. Disponível em: <https://www.hashicorp.com/blog/atlas-announcement>. Acesso em 14 fev 2018.

DIEDRICH, Cristiano. O que é Docker? Disponível em: <https://www.mundodocker.com.br/o-que-e-docker/>. Acesso em 14 fev 2018.

COSTA, Marcelo. Ansible: uma nova opção para gerenciamento de configuração. Disponível

em: <https://www.infoq.com/br/news/2013/04/ansible1.1>. Acesso em 14 fev 2018.