ALCIDES MAYA – FACULDADE E ESCOLA TÉCNICA
CURSO DE TECNÓLOGO EM REDES DE COMPUTADORES

Renã Freitas da cruz

INFRAESTRUTURA POR CÓDIGO - UMA ALTERNATIVA PARA GERENCIAMENTO DE INFRAESTRUTURA DE REDES DE COMPUTADORES.

Porto Alegre
2020

Renã Freitas da Cruz

Trabalho de conclusão de curso de graduação apresentado a Alcides Maya – Faculdade e Escola Técnica como requisito parcial para a obtenção do título de Tecnólogo em Redes de Computadores.

Orientador: Fagner Coin Pereira
Coorientador: Joao Moreira

Porto Alegre

2020

RESUMO

Atualmente muitas empresas do mercado, passam por um momento de necessidade em inovação digital continua para conseguirem se manter competitivas quando comparadas a outras empresas do mesmo segmento, sendo assim é necessário que se tenha uma infraestrutura de redes de computadores que esteja pronta para se adaptar a esse crescimento estrutural que ocorre dinamicamente. É muito comum hoje nas empresas, quando se tem a necessidade de se adicionar uma nova ferramenta em um site ou sistema, em que os responsáveis pelo desenvolvimento abram uma solicitação para o time de Infraestrutura disponibilizarem um ambiente parecido com o que há em produção. Assim possibilita que sejam feitos testes com a mudança que estão trabalhando, e o que irá acontecer quando aplicar em produção. O problema que se encontra nesse modelo é a demora entre a comunicação e a falta de precisão para um ambiente de homologação igual ao que já está em produção aumentando a possibilidade de criar divergências no resultado esperado quando houver a implementação no ambiente de produção, sendo também um agravante os ambientes que precisam ser refeitos por algum erro na parte de teste. Há também os casos onde há a necessidade de realizar a migração de ambientes locais (servidores da empresa) para a hospedagem na nuvem. Nesses casos, geralmente é criado um servidor nesse ambiente externo e instalado os mesmos serviços no outro ambiente, depois feito a restauração das configurações do sistema de produção. Isso também gera uma demora muito grande e está aberto a falhas pois caso algum software esteja com versão desatualizada ou diferente da que hoje está em produção nos servidores, pode influenciar diretamente na funcionalidade do sistema ou até em alguma parte específica. Diminuirmos os problemas acima, mostrando uma "planta" de rede que irá utilizar a ideia de infraestrutura por código, onde todos os servidores serão segmentados de forma padronizada e automatizando a instalação para quando houver a necessidade de se criar ambientes de homologação com apenas a execução de um script se tenha um local para testes que será idêntico ao de produção, diminuindo a possibilidade de resultados inesperados. Podemos também extrair da possibilidade de agilizar e deixar mais seguro a migração de ambientes físicos para nuvem, pois conseguimos replicar de modo idêntico o ambiente de produção para o provedor de serviço da nuvem.

Palavras-chave: Infraestrutura por código, IaC, Linux, Bash, Open Source.

ABSTRACT

Currently, many companies in the market are going through a time when the need for digital innovation continues to remain competitive when compared to other companies in the same segment, so it is necessary that they have a computer network infrastructure that is ready to adapt to this structural growth that occurs dynamically. Today, it is very common in companies, when it is necessary to add a new tool on a website or system, in which users execute the development after a request for available Infrastructure time in an environment similar to the one in production. So it is possible that tests are made with changes that are being carried out and that occur when applied in production. The problem found in this model is the delay between communication and the lack of precision for an approval environment equal to that already in production, making it possible to create divergences in the expected result when there is an execution in the production environment, being it also aggravates the environments that need to be redone due to an error in the test part. There are also cases where there is a need to create local environments (company servers) for hosting in the cloud. In these cases, a server is usually created in this external environment and the same services are installed in the other environment, after restoring the production system settings. This also generates a very long delay and is open to occur if any software is outdated or different from the one currently in production on the servers, it can directly influence the customization of the system or some specific part. Reduce the above problems by showing a network "blueprint" that will use a code infrastructure idea, where all servers will be segmented in a standardized and automated way for installation when there is a need to create approval environments with just one script execution, if there is a testing site that will be identical to production, lessening the possibility of unexpected results. It is also possible to extract the possibility to streamline and make the use of physical environments for cloud safer, since it can replicate the identical mode or the production environment for the cloud service provider.

Keywords: Infrastructure by code, IaC, Linux, Bash, Open Source.

Sumário

Lista de figura

1 - INTRODUÇÃO

A evolução que a tecnologia promove, realiza diversos impactos na população e em resultado disso várias empresas modernizarem seus processos diários implementando melhorias contínuas para acompanhar o mercado de trabalho cada dia mais competitivo. Sendo assim, se torna necessário que a tecnologia da informação (TI) de uma companhia seja uma ferramenta facilitadora de novos ganhos tanto quanto em fatores econômicos quanto na agilidade de processos. O intuito deste trabalho é apresentar a possibilidade de uma rede de computadores ser automatizada ao ponto de conseguir administrar novos ambientes sem que seja necessário muito mais que a execução de um código e que consiga ser escalável para quando houver a necessidade de se expandir ambientes sem causar instabilidade e poder apoiar a empresa no processo de crescimento.

Quando pensamos hoje em grandes empresas que oferecem um valor significativo de inovação, pensamos logo em empresas de tecnologia como por exemplo Facebook, Twitter, Adobe, etc. Além da alta complexidade que há por trás delas, o que há de comum é a velocidade com que elas implementam melhorias diárias, como por exemplo, a Amazon que em média aplica uma nova atualização a cada 11,7 segundos Christopher Null (2019). Com um número tão grande de atualizações diárias pode não parecer palpável fazer todo esse processo sem que que ocorra indisponibilidades de serviço ou da ferramenta nem garantir a integridade do código como um todo.

O suporte a essas atualizações contínuas é possível graças a um processo conhecido como Infrastructure as Code (IaC), com isso um time de desenvolvimento que é responsável por escrever em código as melhorias diretas para uma funcionalidade de um sistema consegue subir ambientes locais ou na nuvem idênticos aos que já tem em produção para testar seu código e identificar possíveis problemas e só após isso disponibilizarem ao público. Iac é uma abordagem para automação de infraestrutura baseada em práticas de desenvolvimento de software. Ele enfatiza rotinas consistentes e repetíveis para aprovisionar e alterar o sistema e sua configuração. As alterações são feitas nas definições e depois lançadas no sistema por meio de processos autônomos que incluem validação completa, conforme Kief Morry (2016).

1.1 Justificativa

Este tema torna se necessário pois no momento em que vivemos onde as mudanças disruptivas acontecem rapidamente e a competitividade empresarial é muito grande as redes de computadores não podem ser um limitador para evolução da empresa.

Hoje precisamos gastar menos tempo no gerenciamento de TI e mais tempo em inovação. Tendo em mente que podemos deixar a manutenção da infraestrutura e concentrar em seu principal objetivo, os negócios da empresa.

Além das situações já citadas é necessário entendermos como realidade a tecnologia de infraestrutura como código pois diversas grandes empresas como adobe, Amazon, Facebook, Netflix, Sony e Walmart já utilizam esse método e se quisermos nos manter competitivos no mercado de trabalho, precisamos estar atentos a como as grandes instituições estão gerando resultados positivos na nossa área.

1.2 Delimitações do Tema

Neste trabalho serão abordados somente os quesitos que tangem uma melhora para um padrão muito comum nas redes de computadores, evitando problemas que derivam da falta de automatização e escalabilidade de ambientes virtuais, permitindo que a estrutura seja mais flexível. Será demonstrado uma infraestrutura de redes de computadores num contexto de uma empresa fictícia que passará por um processo de automatização na implementação de uma ferramenta que se chamado GLPI, demonstrando como a utilização do Terraform pode ser implementada para diversas situações que partem desde a necessidade de migração de uma estrutura localizada em servidores físicos para servidores na nuvem quanto para criar ambientes virtuais idênticos aos que já estão em produção, processo feito para que seja possível realização de testes de linhas de códigos para execução em computadores.

1.3 Objetivos

Os objetivos deste trabalho dividem-se em Objetivo Geral e Específicos.

1.3.1 - Objetivo Geral

Demonstrar como usar a automatização da infraestrutura por código (*infrastructure as code*) pode diminui o tempo gasto para gerenciar de uma infraestrutura de redes em uma empresa contrariando o modelo tradicional que é aplicado de modo comum ondes quase todo processo é manual.

1.3.2 - Objetivos Específicos

- Demonstrar o esquema de funcionamento geral a infraestrutura de redes de computadores.
- Mostrar os principais pontos vulneráveis a falhas críticas, que podem deixar a empresa com seus recursos comprometidos.
- Avaliar as mudanças positivas que uma infraestrutura por código pode trazer para redes diminuindo pontos de falhas.
- Exemplificar um processo automatizado de umas publicações (*deploy*) de um sistema WEB.
- Mostrar os pontos positivos e negativos do processo.

2 REFERENCIAL TEÓRICO

Nessa seção serão apresentados os principais conceitos para embasamento do trabalho em desenvolvimento.

2.1 Infraestrutura como código

Infraestrutura como código é um modo de gerenciamento e provisionamento de configuração que utiliza arquivos de configuração escrito através de uma linguagem de programação simples e declarativa ao invés de configurações manuais feitos tanto no ambiente físico como virtual, que segundo Nelsom-Smith (2013), o poder de computação infinita combinado com a urgência de se criar grandes ambientes de produção, permite que todos que queira criar um website dinâmico passem pela necessidade de deixar que fique facilmente escalável, processo esse que é uma das propostas do IaC.

2.2 Servidor WEB apache

Criado em 1995 por Rob McCool, que trabalhava NCSA (National Center for Supercomputing Applications), o servidor Apache é o mais bem reconhecido servidor para aplicações web de código livre que existe, utilizado principalmente em Linux.

Conforme Marcelo e Antonio (2005), o servidor Web Apache é um dos softwares mais utilizados nos dias de hoje na Internet. O Apache é sem sombra de dúvidas um dos mais robustos e seguros programas desenvolvidos para ambientes e que mantém mais de 60% das homepages disponíveis do mundo.

2.3 Central de serviços (GLPI)

Conforme Sales(2014), a medida que as empresas e seu profissionais amadurecem, estão aprendendo que a chave do sucesso do seu negócio está na eficiências dos seus processos. Para auxiliar nas situações acima, temos o sistema chamado GLPI que um sistema web de gerenciamento de ativos de TI e *Service Desk* gratuito, escrito numa linguagem de programação conhecida como PHP e distribuído como General Public Licence (GNU) que permite que ele seja estudado, executado e modificado sem maiores restrições.

2.4 Software Livre - Ubuntu

O Ubuntu é um sistema operacional que começou a ser desenvolvido em 2004 pela empresa Canonical, fundada pelo sul-africano Mark Shuttleworth, Como o núcleo Linux apresenta o seu código fonte sob a licença GPL (Licença Pública Geral). Inclusive, concordante SILVEIRA (2004) esse tipo de software livre é um movimento pelo compartilhamento do conhecimento tecnológico e começou nos anos 1980 e se espalhou pelo planeta.

2.5 Serviço em Nuvem - AWS

O Amazon Elastic Compute Cloud (Amazon EC2) oferece uma capacidade de computação redimensionável na nuvem da Amazon Web Services (AWS). O uso do Amazon EC2 elimina a necessidade de investir em hardware inicialmente, portanto, você pode desenvolver e implantar aplicativos com mais rapidez. É possível utilizar o Amazon EC2 para executar o número de servidores virtuais que precisar, configurar a segurança e a rede, e gerenciar o armazenamento, conforme informado pela própria documentação da Amazon AWS (2010).

2.6 Software de IaC - Terraform

O Terraform é uma ferramenta para criar, alterar e criar versões de infraestrutura com segurança e eficiência. Este mecanismo pode gerenciar provedores de serviços existentes e populares, bem como soluções personalizadas internas. Os arquivos de configuração descrevem para Terraform os componentes necessários para executar um único aplicativo ou todo o seu datacenter. O Terraform gera um plano de execução descrevendo o que fará para atingir o estado desejado e, em seguida, executa-o para construir a infraestrutura descrita. Em decorrência de mudanças, o Terraform pode determinar o que mudou e criar planos de execução incrementais que podem ser aplicados HashiCorp (2015).

2.7 Hyperv

O Hyper-V permite executar vários sistemas operacionais como máquinas virtuais no Windows. Especificamente, o Hyper-V fornece a virtualização de hardware. Isso significa que cada máquina virtual é executada em hardware virtual. O Hyper-V permite que você crie discos rígidos virtuais, comutadores virtuais e um número de outros dispositivos virtuais que podem ser adicionados a máquinas virtuais Microsoft (2014).

2.8 PHP

Hypertext Preprocessor (PHP) é uma linguagem de código aberto de *scripts* de propósito geral, especialmente preparada para o desenvolvimento Web e pode ser incorporado à códigos HTML. Segundo Barreto (2000), o que diferencia o PHP de outras linguagens como Javascript é que o código PHP é executado somente no servidor, gerando HTML puro para ser enviado para o cliente (quem requisitou acesso ao site). Sendo uma vantagem não expor o código fonte.

3 AMBIENTES DE DESENVOLVIMENTO

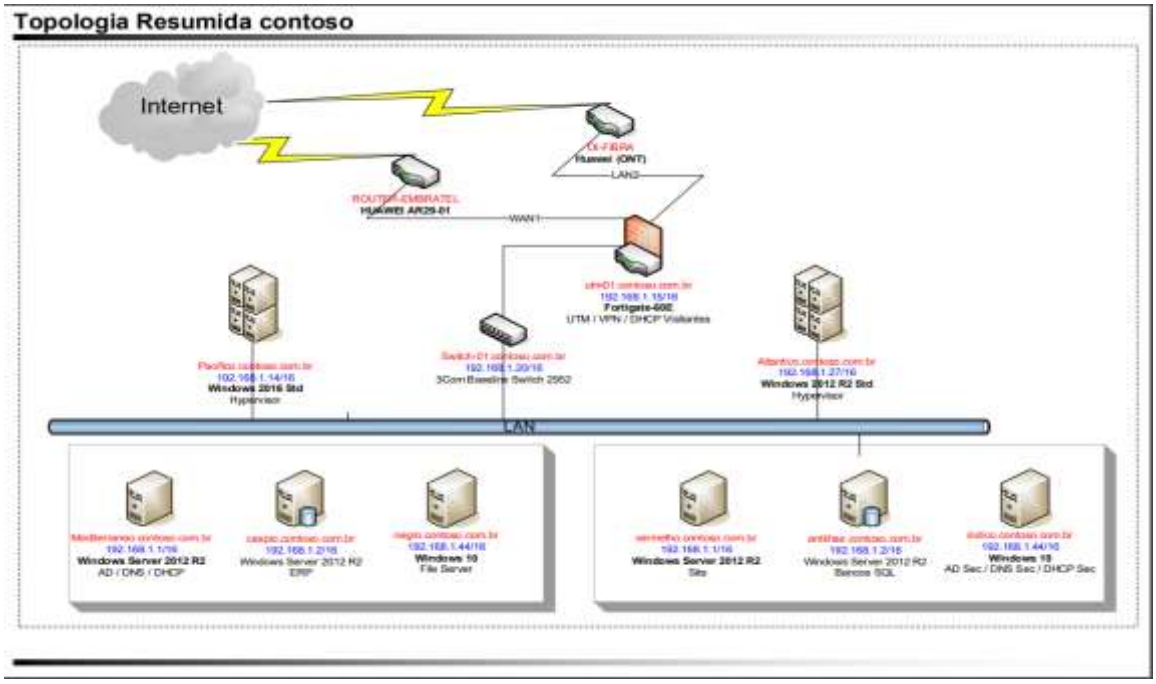
Atualmente as empresas que querem possuir um diferencial competitivo para ficarem em destaque e atrair mais clientes, precisam ser capazes de além de atender as necessidades básicas do cliente, também oferecem serviços além do esperado, aumentando sua cota de mercado.

Esse aperfeiçoamento no modo como a empresa se dispõe ao cliente necessita de um ambiente estável e com credibilidade de funcionamento que seja de fácil expansão em caso de aumento de demanda. Utilizando o exemplo de uma empresa de varejo que desenvolveu seu aplicativo para smartphone (APP) e precisa estar sempre disponível, até em momentos de alta demandas de compras, como ocorre no feriado de natal que as vendas passaram por uma alta no volume de 29,9% no ano de 2019 comparado a 2018 (ECBR, 2019) .

A partir da necessidade de um ambiente virtual com alta disponibilidade computacional, que não esteja dependendo de serviços centralizado e escaláveis, surge um modo mais simples de se gerenciar processos em redes de computadores que é a IaC, onde diferente do modo tradicional de processos feitos todos manuais ele permite que com a execução um código no computador que é executado através do interpretador de comandos seja criado um ambiente virtual inteiramente novo ajustado às novas demandas que são solicitadas.

Para que seja possível fazer um comparativo entre os processos de gerência, foi criado um ambiente que se assemelha ao que poderia ser encontrado num ambiente de produção de muitas empresas, em acordo a figura 1.

Figura 1- Funcionamento de uma rede tradicional



Fonte: Elaborado pelo autor

É possível identificar ao analisarmos a figura 1 uma empresa nomeada como “contoso” onde sua rede é dividida do seguinte modo: Dois servidores Físicos (Pacifico e Atlantico) que cumprem o papel de virtualizadores e utilizam o sistema de virtualização chamado Hyper-v e 6 servidores virtuais (Mediterraneo, caspio, negro, vermelho, antilhas, indico). Cada servidor tendo seu papel na infraestrutura, as máquinas mediterrâneo e indico recebem os principais serviços para a rede LAN que são, o AD (*active directory*) responsável pela autenticação de usuários em suas máquinas, caspio que recebe o sistema da empresa, o negro com os arquivos de redes onde os usuários trabalham, antilhas com papel de receber o banco de dados SQL e o vermelho responsável pelo site acessado externamente da empresa.

O problema em mantermos as redes de computador de forma estática é que o ambiente vira um limitador para a capacidade da empresa e traz possíveis pontos de falhas, um exemplo é caso seja necessário desenvolver uma nova ferramenta para o site e precise se criar um ambiente de testes antes de aplicar as mudanças. Quando dividimos o ambiente de trabalho entre produção e homologação, é preciso garantir que durante a realização de testes não haverá variáveis de sistema, precisamos garantir que o servidor de produção será igual a homologação.

3.1 Ambiente de testes

Para o ambiente de testes é utilizado um virtualizador de máquinas virtuais chamado de Hyper-v, ele irá receber um sistema operacional Ubuntu que será responsável por receber o software Terraform. Com a estrutura virtualizada criada e respectivo software instalado, será criado os códigos necessários para simularmos um ambiente onde uma empresa precisa fazer uma migração escalável de seu site localizado localmente nos servidores para um serviço de nuvem AWS.

4 DESENVOLVIMENTO

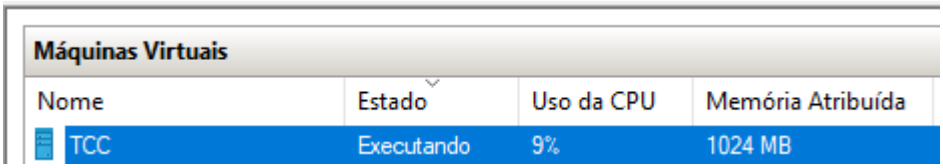
Nessa seção serão apresentados os pontos a qual o trabalho se propôs demonstrando a ferramenta Terraform como uma alternativa mais rápida para gerenciamento de infraestruturas de redes de computadores.

4.1 Criando Ambiente Tradicional

Conforme citado, nesse primeiro momento foi montado um cenário comum onde o servidor de virtualização local é utilizado para receber e dividir os serviços comuns na rede, para isso iremos utilizar a ferramenta Hyper-v.

Para isso, foi realizado a configuração de uma máquina virtual utilizando um processador virtual, 1gb de memória ram e aproximadamente 60gb de espaço em disco.

Figura 2- Configuração da VM - Hyperv



Fonte: Elaborado pelo autor

Neste servidor, é operado o sistema operacional Ubuntu 18.04, instalando o apache 2.4 e o sistema de centralizador de chamado GLPI na versão estável 9.2. Após configuração inicial da ferramenta web GLPI o funcionamento está estável, conforme pode ser visto na figura 3.

Figura 3- Funcionamento do sistema GLPI

Fonte: Elaborado pelo autor

Por questões de boas práticas em desenvolvimento de softwares, as aplicações são separadas do servidor de banco de dados, portanto novamente no Hyper-v é configurado outro servidor virtual, seguindo a mesma configuração anterior que era um processador virtual, 1gb de memória ram e aproximadamente 60gb de espaço em disco. Diferente do primeiro servidor, neste apenas instalamos o banco de dados mariadb 10.0.38.

4.2 Instalação de software Terraform

O software Terraform além de ser uma ferramenta altamente versátil está disponível em várias plataformas, como por exemplo MacOS, FreeBSD, OpenBSD, Solaris, Windows e Linux.

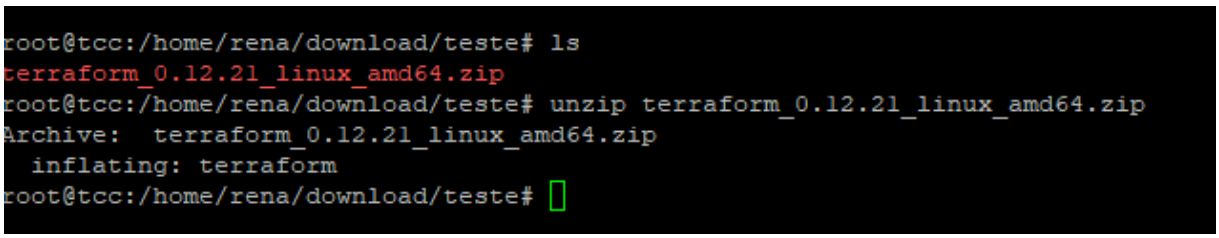
Como no laboratório que foi proposto é utilizado o sistema operacional Ubuntu que é uma distribuição Linux, iremos utilizar a versão recomendada para instalação do Terraform que é a 0.12. Para isso precisamos acessar o servidor utilizando uma ferramenta nativa do sistema operacional que permite a execução de comandos através de texto, conforme figuras que seguem.

Figura 4- Realizando download do Terraform

Fonte: Elaborado pelo autor

Após realizado o download do arquivo é necessário deixar ele pronto para execução, como por padrão ao realizar o *download* do Terraform ele se encontra numa forma comprimida, será necessário extrairmos os arquivos para outro diretório, para isso é executado o comando demonstrado na figura 5.

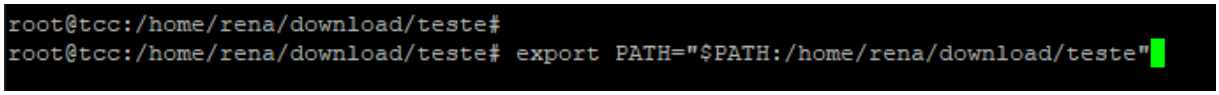
Figura 5- Ajuste pós instalação do Terraform



Fonte: Elaborado pelo autor

Logo feito o processo conforme figura 5, o Terraform já está apto para utilização e para facilitarmos a sua execução no sistema operacional foi necessário um ajuste onde informamos que ao executar a palavra “Terraform” no interpretador de comandos o sistema deveria executar o software que conforme figura 5 extraímos para /home/rena/download/teste, o comando essencial para tornar isso possível se encontra na figura 6.

Figura 6- Parametrização de atalhos em Terraform



Fonte: Elaborado pelo autor

4.3 Instalação simplificada utilizando o Terraform

A automatização do Terraform funcionando através de um arquivo de texto que deve ser salvo com a extensão .tf num estilo conhecido como "*cookbook*" que é uma expressão em inglês para "livro de receita" onde você cita através de passos o que a ferramenta deverá fazer.

Figura 7- Instalação de servidor com Terraform

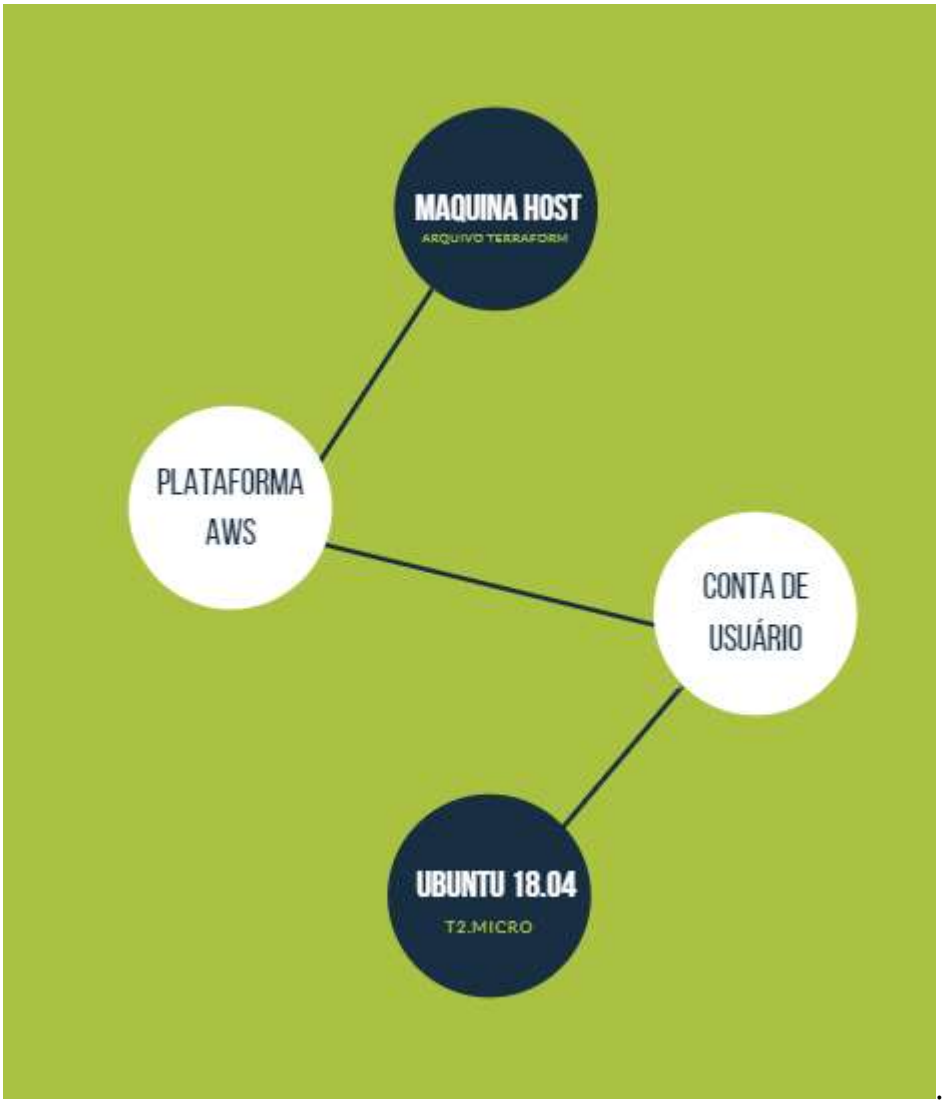
```
provider "aws" {
acess_key = "AKIAJEHMMCWOQIRDDJS3"
secret_key = "uEjkmsFjsn2SwkpoemhbahsuwombfdhWsklksW231Sd"
region = "us-east-2"
}

resource "aws_instance" "Teste_terraform" {
        ami = "ami-0f65671a86f061fcd"
        instance_type = "t2.micro"
}
```

Fonte: Elaborado pelo autor

Na imagem da figura 7, estamos utilizando o estilo *cookbook*, descrevendo como o Terraform deverá fazer a configuração do ambiente. Para essa situação o Terraform criou uma instância virtual na AWS com o nome de "Teste_Terraform" utilizando um sistema operacional do Ubuntu na versão 18.04 no tipo de instância t2.micro da AWS, fazendo com que o ambiente virtual que está na nuvem seja idêntico ao que foi criado virtualmente através do Hyper-V.

Figura 8- Organograma de funcionamento da instalação utilizando Terraform

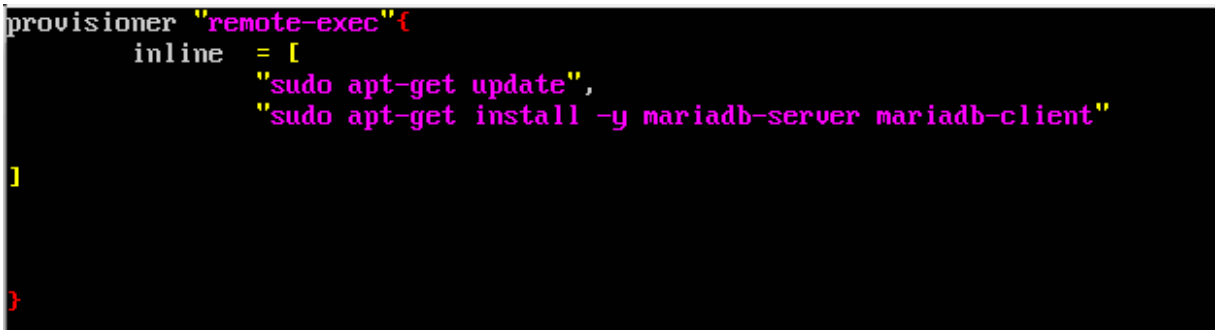


Fonte: Elaborado pelo autor

4.4 automatizando instalação de ambiente para nuvem

Conforme mostrado na figura 8, pode ser visto que com aproximadamente 10 linhas, podemos automatizar a instalação básica de um sistema operacional, porém a grande vantagem que o Terraform nos possibilita é automatizar instalação de serviços utilizando a mesma estrutura de código

Figura 9- Utilizando remote-exec para parametrização de ferramentas



Fonte: Elaborado pelo autor

Concordante a figura 9, é utilizando a linha de código para atualização do sistema operacional e realizar a instalação do serviço que formará o banco de dados da aplicação, que no caso será utilizado o mariadb, porém a um outro modo onde podemos personalizar a instalação e deixar ela com os parâmetros necessário para aumentar o nível de automatização. Uma das ferramenta do Terraform é o "File Proviosiner", que conforme a documentação permite que seja feito uma cópia de arquivos/diretórios da máquina host (está executando o Terraform) para a local de destino. Um exemplo prático para o uso disso está abaixo na figura 10.

Figura 10- Utilizando remote-exec para parametrização de ferramentas

Fonte: Elaborado pelo autor

Convergente ao a figura 10, foi realizado a utilização do parâmetro "remote-exec", para após o processo de criação dos servidores na estrutura da AWS o serviço Terraform execute a instalação do serviço GLPI. A ordem de funcionalidade da ferramenta até aqui está descrita na figura 10.

Figura 11- Instalação automatizada utilizando Terraform

Fonte: Elaborado pelo autor

Seguindo o propósito da figura 10 a máquina host, que recebe o serviço do Terraform inicia o processo de instalação de um novo servidor utilizando a plataforma da AWS, após realizar as devidas autenticações com sua conta de usuário, é configurado um servidor com sistema operacional Ubuntu 18.04, e o Terraform executar o script “install_glpi” que conforme a figura 9 demonstra, realiza a atualização do sistema operacional, instalar o banco de dados necessário.

5 METODOLOGIA

Para este trabalho foi realizado uma pesquisa bibliográfica utilizando de inúmeras bases de referências tais como livros técnicos direcionados às tecnologias utilizadas neste trabalho e a documentação oficial das ferramentas utilizadas que estão disponíveis nos sites dos responsáveis por manter os softwares em funcionamento.

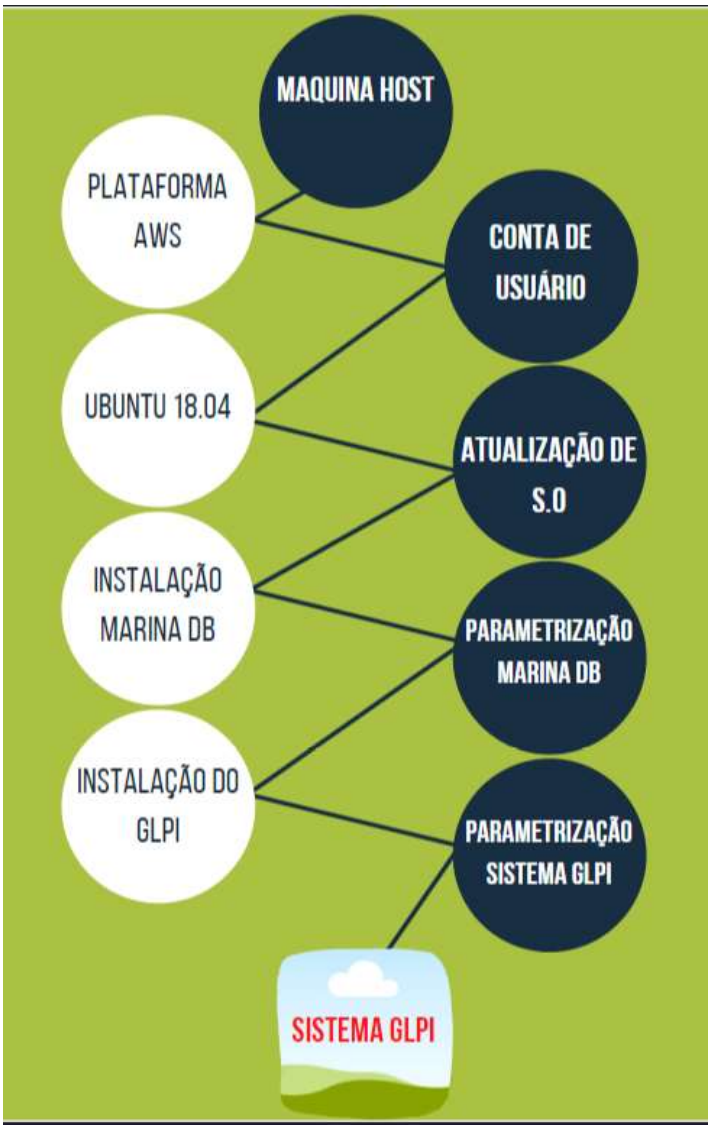
Foi realizado uma aplicação prática num ambiente de laboratório virtual demonstrado um cenário onde todo procedimento de instalação de servidor, atualização de pacotes de softwares e sistema operacional, configuração de softwares de dependência e parametrização do banco de dados manualmente para o funcionamento do sistema de gestão de ativos e helpdesk conhecido como GLPI, após isso utilizamos a ferramenta Terraform com o intuito de automatizamos todo o processo a fim de facilitar, evitar erros e diminuir o tempo necessário para o sistema ficar acessível, com esse processo também ficou claro como e possível automatizar outros processo de instalação abrindo possibilidade para casos onde é preciso replicar ambientes para se usar base de testes, migrar sistemas locais para nuvem ou automatizar atualizações de software em tempo real.

Os dados de tempo na aplicação das duas propostas foram coletados através da execução do escopo proposto durante o trabalho num ambiente idêntico aos dois modos anotando os tempos de execução das tarefas e depois comparando aos resultados obtidos através da execução do script de automatização da rotina além do apontamento de possíveis falhas que ficam óbvias que a utilização do modo manual pode trazer ao dia a dia da empresa.

6 RESULTADOS

Para ser realizado os testes comparativos foi replicado a configuração que fizemos previamente utilizando o modo automatizado com o Terraform concordante a figura 10 para o modo tradicional que é feito manualmente, está com seu processo exemplificado na figura 11.

Figura 12- Instalação do GLPI sem automatização do Terraform



Fonte: Elaborado pelo autor

Demonstrado na figura 11, pode ser observado que o processo manual sem a utilização do Terraform envolve aproximadamente 8 etapas que são o acesso a plataforma da AWS, autenticação com o seu usuário, criação de uma instância virtual utilizando o sistema operacional Ubuntu 18.04, acesso a instância criada e a atualização do sistema operacional,

instalação do banco de dados que é o Marina DB, fazer as devidas configurações no banco de dados, como criação de bases e ajustes em senhas de acesso, instalar o sistema GLPI, parametrizar as configurações necessárias até que enfim o sistema GLPI consiga estar disponível para acesso. Diferentemente da figura 10 onde o mesmo processo é exemplificado, nessa situação o processo de instalação de máquina virtual, configuração da ferramenta, atualização de sistema e devidos ajustes padronizados nos acessos foi identificado o dobro de procedimentos necessários.

Durante a utilização da ferramenta Terraform para automatização e o modo manual foi possível identificar uma diferença significativa no tempo necessário para configuração do mesmo ambiente com essas duas formas distintas. Nos testes realizados usando o Terraform, o resultado foi de um tempo médio de aproximadamente 15 minutos para deixarmos a aplicação acessível. Durante a execução do mesmo processo sendo feito manualmente, foi tido um resultado de aproximadamente 40 minutos. Um fator importante que somou na demora da aplicação ficar acessível utilizando o método manual foi o fato de um erro de digitação durante a configuração do banco de dados que seria utilizado pelo GLPI, o que gerou o tempo extra necessário para identificação e conserto do erro e após isso podermos dar continuidade nas devidas configurações.

7 CONSIDERAÇÕES FINAIS

Neste trabalho foi demonstrado um processo relativamente simples que é muito comum no trabalho para diversas empresas com diversos tamanhos e cultura e que pode se tornar um problema no dia a dia, já que a há necessidade de processos responderem de uma forma mais automatizada e sem espaços para falha ou descuidos. Essa situação se torna mais importante quando se é levado em conta a velha premissa de que “tempo e dinheiro”.

Como uma solução para esse problema é apresentado o software Terraform que e um automatizador de configurações para estruturas e configurações de software, neste trabalho ele é utilizado num contexto onde uma empresa denominada “contoso” precisa realizar a instalação de um sistema para controle de atendimentos e helpdesk conhecido como GLPI e que ao utilizar o modo manual de instalação manual precisou do dobro do tempo para o software estar disponível para acesso quando comparado com a utilização do Terraform.

A importância desse tema fica clara quando se transporta o cenário apresentando de utilização para uma aplicação de uma grande corporação como por exemplo o Facebook que a implementação de mudanças funciona a cada 11,7 segundos.

Para processos que são tão comuns e que ocorrem diariamente nessas corporações não se pode depender de um processo manual onde um erro humano pode causar indisponibilidade na ferramenta ou todo sistema o que pode causar uma perda significativa na receita financeiro.

Como a aplicação desse trabalho foi realizado num ambiente de laboratório virtual, foi possível fazer uma réplica do mesmo processo utilizando o Terraform e o modo manual de execução do processo e com isso foi identificado um tempo de aproximadamente 40% menor ao utilizar a ferramenta de automatização Terraform e ao realizar outros testes de replicação não tivemos nenhum erro de execução já que a ferramenta estava parametrizada corretamente e funcionando.

A execução desta monografia permitiu demonstrar todos os pontos que haviam sido propostos, citando a ideia de demonstrar o esquema de funcionamento geral a infraestrutura de redes de computadores com uma aplicação proposta numa empresa chamado “contoso” que está na figura 1.

Foi possível também demonstrar os principais pontos vulneráveis a falhas críticas,

que podem deixar a empresa com seus recursos comprometidos quando se apresenta a necessidade de diversas atualizações num tempo muito curto e um *deploy* de aplicações como sistema GLPI num ambiente que não foi testado previamente ou foi testado num ambiente que não se iguala ao de produção que pode gerar erro no sistema que estará disponível para o usuário final.

CRONOGRAMA

Atividade	Meses											
	Ano 2019						2020					
	Jul	Ago	Set	Out	Nov	Dez	Fev	Mar	Abr	Mai	Jun	Jul
Escolha do assunto do projeto	x											
Elaboração da estrutura do projeto	x	x	x									
Seleção e Leitura das obras para elaboração do projeto			x	x	x							
Elaboração dos objetivos, delimitação do tema, definição do problema, etc			x	**x**								
Elaboração da pesquisa bibliográfica e documental do projeto.				x								
Coleta de dados				x	x	x	x	x				
Tratamento dos Dados							x	x				
Revisão final do texto e elaboração da introdução e conclusão								x	x	x	x	
Data limite de entrega do projeto de estágio												x

REFERÊNCIAS

AWS Documentation **O que é o Amazon EC2**, 2018. Disponivel em: <https://docs.aws.amazon.com/pt_br/AWSEC2/latest/UserGuide/ec2-ug.pdf#concepts>. Acessado em 08 de setembro de 2019.

BACON, Jono. **O Livro oficial do Ubuntu**. São Paulo: Bookman, 2008.

Barreto, Mauricio Vivas de souza. **Curso de linguagem PHP, 2000.** Disponivel em: <http://www.etelg.com.br/paginaete/downloads/informatica/php.pdf>. Acessado em 01 de março de 2020.

E-commerce brasileiro fatura R$14,1 bi no Natal e evita R$ 292,5 mi em prejuízos com fraudes, 2019. Disponivel em: <https://www.ecommercebrasil.com.br/noticias/e-commerce-fatura-14-bi-no-natal-2019/>. acessado em 27 de abril de 2020.

HashiCorp Documentation **What is Terraform?**, 2015. Disponivel em: <https://www.Terraform.io/intro/index.html>. Acessado em 21 de fevereiro de 2020.

NELSON-SMITH, Stephen. **Test-Driven Infrastructure.** Califórnia, EUA: O'Reilly Media, 2013.

MARCELO, Antonio. **Apache:** configurando o servidor WEB. Rio de Janeiro, Brasil: Brasport, 2005.

Microsoft Documentation **Introdução ao Hyper-V no Windows 10**, 2014. Disponivel em: <https://docs.microsoft.com/pt-br/virtualization/hyper-v-on-windows/about/>. Acessado em 21 de fevereiro de 2020.

Morry, Kief. **Infrastructure as Code: Managing Servers in the Cloud. .** São Paulo: O'Reilly Media, 2016.

Null, Christopher. **10 companies killing it at DevOps, 2019.** Disponivel em: <https://techbeacon.com/devops/10-companies-killing-it-devops>. acessado em 27 de abril de 2020.

SILVEIRA, Sérgio Amadeu da. **Software Livre: luta pela liberdade do conhecimento**. São Paulo: Fundação Perseu Abramo, 2004.

SALES, Halexsandro de Freitas **Central de Serviços com Software Livre: Estruturando uma Central de Serviços com o GLPI.** São Paulo: Edição do Autor, 2014.