UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

PAULO MENDES DA SILVA JÚNIOR

ESTUDO COMPARATIVO ENTRE FERRAMENTAS DE AUTOMAÇÃO PARA IMPLANTAÇÃO DE INFRAESTRUTURA EM TI

CAMPINA GRANDE - PB

2022

PAULO MENDES DA SILVA JÚNIOR

ESTUDO COMPARATIVO ENTRE FERRAMENTAS DE AUTOMAÇÃO PARA IMPLANTAÇÃO DE INFRAESTRUTURA EM TI

Trabalho de Conclusão de Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Professor Dr. Reinaldo Cézar de Morais Gomes

CAMPINA GRANDE - PB

2022

PAULO MENDES DA SILVA JÚNIOR

ESTUDO COMPARATIVO ENTRE FERRAMENTAS DE AUTOMAÇÃO PARA IMPLANTAÇÃO DE INFRAESTRUTURA EM TI

Trabalho de Conclusão de Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA:

Professor Dr. Reinaldo Cézar de Morais Gomes

Orientador – UASC/CEEI/UFCG

Professor Dr. Leandro Balby Marinho

Examinador – UASC/CEEI/UFCG

Professor Dr. Tiago Lima Massoni

Professor da Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em: 06 de Abril de 2022.

CAMPINA GRANDE - PB

ABSTRACT

The specification and evaluation of software quality and software-intensive computer systems is a key factor to ensuring value to stakeholders. So that, it is important that quality characteristics are specified, measured and evaluated whenever possible, using validated and widely accepted measures and methods. Regarding the Infrastructure as Code (IaC) topic, there are no studies that compare the tools for managing automation configuration in software, in terms of well-defined metrics, which are used as requirements for the evaluation of a good quality of software, such as the metrics defined and documented by ISO/IEC 25010:2011 [1]. By analyzing all metrics defined in the specification, a subset was chosen to analyze some infrastructure automation tools. The metrics chosen were: maintainability, portability and usability. It was observed that all tools play the role of infrastructure configuration as code well, but some stand out for their simplicity and easy learning, such as Ansible and Terraform. In addition, Terraform is capable of not only configuring virtual machines, but also provisioning them. On the otherside, Puppet is a little more complex than the other tools because its agent configuration is more complex and the learning curve is not steep, but it also works very well for infrastructure configuration.

Estudo comparativo entre ferramentas de automação para implantação de infraestrutura em TI

Trabalho de Conclusão de Curso

Paulo Mendes da Silva Júnior
Universidade Federal de Campina Grande
Campina Grande, Paraíba
paulo.junior@ccc.ufcg.edu.br

Reinaldo Cézar de Morais Gomes
Universidade Federal de Campina Grande
Campina Grande, Paraíba
reinaldo@computacao.ufcg.edu.br

RESUMO

A especificação e avaliação da qualidade de software e dos sistemas de computador com uso intensivo de software é um fator chave para garantir valor às partes interessadas. Para isso, é importante que as características de qualidade sejam especificadas, medidas e avaliadas, sempre que possível, usando medidas e métodos validados e amplamente aceitos. Em relação ao tema Infrastructure as Code (IaC), não existem estudos que fazem um comparativo entre as ferramentas para gerenciamento de configuração de automação em software, em termos de métricas bem definidas, que são usadas como requisitos para a avaliação de uma boa qualidade de software, como por exemplo, as métricas definidas e documentadas pela ISO/IEC 25010:2011 [1]. Ao analisar todas as métricas definidas na especificação, um subconjunto foi escolhido para analisar algumas ferramentas de automação de infraestrutura. As métricas escolhidas foram: manutenibilidade, portabilidade e usabilidade. Foi observado que todas as ferramentas desempenham bem o papel de configuração de infraestrutura como código, porém algumas se destacam pela simplicidade e fácil aprendizado, como é o caso do Ansible e do Terraform. Além de que o Terraform é capaz de não somente configurar máquinas virtuais, mas também provisioná-las. Por outro lado, o Puppet é um pouco mais complexo do que as demais ferramentas pelo fato de sua configuração de agentes ser mais complexa e a curva de aprendizado não ser acentuada, porém também funciona muito bem para a configuração de infraestrutura.

Palavras-Chave

Automação, Infraestrutura, IaC, Gerenciamento de Configuração, Métricas de qualidade, Estudo comparativo.

1. INTRODUÇÃO

O provisionamento da infraestrutura sempre foi um processo manual, o que o torna demorado e caro.
Com a computação em nuvem, o número de componentes da infraestrutura aumentou, mais aplicativos estão sendo lançados para produção diariamente e a infraestrutura precisa ser aumentada, escalonada e desativada com frequência. Sem uma prática de *Infrastructure as Code* (ao longo deste trabalho, chamaremos simplesmente pela sigla *IaC*) em vigor, fica cada vez mais difícil gerenciar a escala da infraestrutura de hoje.
Atualmente, o termo *IaC* vem ganhando força no mundo da tecnologia da informação. Utilizando-a é possível gerenciar e provisionar a infraestrutura em TI por meio de código, eliminando a necessidade de processos manuais. Com *IaC*, são criados arquivos de configuração que contém suas especificações de infraestrutura, o que torna mais fácil editar e distribuir configurações.
Implementar uma Infraestrutura como Código também significa que você pode definir sua infraestrutura em componentes modulares que podem ser combinados de diferentes maneiras por meio da automação de software.
A automação do provisionamento de infraestrutura com IaC significa que os desenvolvedores não precisam provisionar e gerenciar manualmente servidores, sistemas operacionais, armazenamento e outros componentes de infraestrutura a cada vez que desenvolvem e implantam um aplicativo.
Dentre os principais benefícios da utilização de IaC para gerenciamento e provisionamento da infraestrutura em TI estão: redução de custos, aumento na velocidade de implantação, redução de erros, melhoria da consistência da infraestrutura e eliminação de desvios de configuração.
Para a avaliação da qualidade de ferramentas para automação de infraestrutura, existem organizações mundiais, como por exemplo: a ISO (Organização

Internacional de Padronização) e a IEC (Comissão Eletrotécnica Internacional), que formam o sistema especializado para padronização mundial de tecnologias elétricas, eletrônicas e relacionadas. Normas internacionais são elaboradas de acordo com as regras dadas nas diretivas da ISO/IEC. A ISO/IEC 25010:2011, mais especificamente, trata de características e subcaracterísticas de qualidade que um software deve garantir para que o mesmo seja considerado um produto de qualidade.

2. O PROBLEMA

A partir de estudos realizados na internet, foi percebido que não existem estudos comparativos entre ferramentas de Automação de Infraestrutura, principalmente, com base em métricas de qualidade definidas pelo sistema especializado para padronização mundial (ISO/IEC).

3. OBJETIVOS

O objetivo deste estudo é comparar ferramentas para automação de infraestrutura em TI (IaC), em termos de métricas da ISO/IEC, mais especificamente, a ISO/IEC 25010/2011, que são métricas definidas pelo sistema especializado para padronização mundial, e que definem características que um software deve possuir para que o mesmo tenha uma boa qualidade.
Para esse estudo, dentre as inúmeras métricas disponíveis na documentação, foram escolhidas: manutenibilidade, portabilidade e usabilidade.
No estudo, serão utilizadas as seguintes ferramentas para automação de infraestrutura: Ansible, Puppet e Terraform, onde cada uma das ferramentas será estudada e avaliada para cada uma das métricas mencionadas anteriormente. Ao final, pretende-se mostrar qual(is) tecnologia(s) melhor se adapta(m) aos padrões de qualidade propostos nesse estudo.

4. CONTEXTUALIZAÇÃO

Nesta seção, apresentaremos informações sobre alguns conceitos e tecnologias que foram utilizadas nesse trabalho, a fim de auxiliar o entendimento do contexto do mesmo.

4.1. Métricas escolhidas para avaliação das ferramentas de automação de infraestrutura (IaC)

Para esse trabalho, foram escolhidas 3 métricas entre as dezenas que estão disponíveis na documentação da ISO/IEC 25010. Todas essas métricas definem atributos de qualidade de software, atributos esses que são relevantes para a construção de um software de qualidade. Todas as métricas e seus respectivos detalhes podem ser facilmente encontrados no site oficial da ISO [1]. Abaixo estão descritas, cada uma das métricas que foram escolhidas para esse estudo.

4.1.1. Manutenibilidade:

Grau de eficácia e eficiência com que um produto ou sistema pode ser modificado pelos mantenedores pretendidos. As modificações podem incluir correções, melhorias ou adaptação do software às mudanças no ambiente e nos requisitos e especificações funcionais. A capacidade de manutenção pode ser interpretada como uma capacidade inerente do produto ou sistema para facilitar as atividades de manutenção, ou a qualidade em uso experimentada pelos mantenedores com o objetivo de manter o produto ou sistema.

4.1.2. Portabilidade:

Grau de eficácia e eficiência com que um sistema, produto ou componente pode ser transferido de um hardware, software ou outro ambiente operacional de uso para outro. A portabilidade pode ser interpretada como uma capacidade inerente do produto ou sistema para facilitar as atividades de portabilidade ou a qualidade em uso experimentada com o objetivo de portar o produto ou sistema.

4.1.3. Usabilidade:

Grau em que um produto ou sistema pode ser usado por usuários especificados para atingir objetivos específicos com eficácia, eficiência e satisfação em um contexto de uso especificado.

4.2. Ferramentas de automação de infraestrutura escolhidas para a avaliação comparativa

Para esse estudo, entre as inúmeras ferramentas para automação de infraestrutura em TI que existem, foram escolhidas três ferramentas que, a partir de pesquisas [10] e opiniões de profissionais que trabalham na área, são comumente utilizadas para automações de infraestrutura em TI. Abaixo temos um breve resumo sobre cada uma das tecnologias escolhidas, onde estas serão avaliadas em função das métricas mencionadas no tópico anterior.

4.2.1. Ansible

O Ansible [2] é uma ferramenta de automação de TI que automatiza o provisionamento em nuvem, o gerenciamento de configurações, a implantação de aplicativos, a orquestração intra-serviços e muitas outras necessidades de TI. Amplamente utilizada por

profissionais de Devops. Ansible utiliza uma linguagem muito simples (Yaml, na forma de Ansible Playbooks) e foi projetado para implantações de várias camadas desde o primeiro momento, modelando sua infraestrutura em TI, descrevendo como todos os sistemas se inter-relacionam, em vez de apenas gerenciar um sistema por vez.
O Ansible funciona conectando-se aos seus nós e executando programas pequenos, chamados "Módulos Ansible". Esses programas foram criados para serem modelos de recursos do estado desejado do sistema. Onde o Ansible executa esses módulos (por SSH, por padrão) e os remove quando finalizados.

4.2.2. Puppet

O Puppet [3] é um utilitário de gerenciamento de configuração de software. Desenvolvido pela Puppet Labs, o Puppet automatiza o processo de entrega de software, permitindo a configuração de máquinas físicas ou virtuais, a orquestração, a emissão de relatórios, ou ainda a distribuição de códigos em fase inicial de desenvolvimento, testes, lançamentos ou atualizações. A arquitetura mais utilizada em conjunto com o Puppet é a de cliente-servidor, por meio do Puppet agent (cliente) e o Puppet master (servidor). No entanto, também é possível utilizar o Puppet independentemente, por meio do Puppet apply.
O Puppet master pode rodar em um ou mais servidores, e é onde os arquivos de configuração ficam armazenados, enquanto que os Puppet agents, geralmente estão configurados para interagir com o Puppet master.

4.2.3. Terraform

Terraform [4] é uma ferramenta de código aberto de IaC criada pela HashiCorp.
Terraform permite aos desenvolvedores usarem uma linguagem de configuração de alto nível chamada HCL (HashiCorp Configuration Language) para descrever a infraestrutura na cloud ou em implementação local de estado final desejada para executar um aplicativo. Em seguida, ele gera um plano para alcançar esse estado final e executa o plano para fornecer a infraestrutura.
Uma vez que o Terraform usa uma sintaxe simples, é possível fornecer a infraestrutura em data centers multi-cloud e locais, sendo possível fornecer novamente a infraestrutura de maneira segura e eficiente em resposta às mudanças de configuração, sendo atualmente uma das ferramentas de automação de infraestrutura mais populares disponíveis, principalmente se o usuário precisa provisionar ambientes de *cloud híbrida* ou *multi cloud*.

5. METODOLOGIA

Para desenvolver este trabalho, foram configuradas/provisionadas máquinas virtuais com cada uma das ferramentas de infraestrutura escolhidas, permitindo a comparação das diferenças entre os resultados de cada uma das execuções, levando em consideração as métricas de qualidade que escolhemos para a avaliação.
Nesse estudo, foram utilizadas três máquinas virtuais previamente criadas: duas com o sistema operacional CentOS 7 e uma com Windows 10. Uma das máquinas CentOS 7 é responsável por enviar e aplicar as configurações nas outras duas máquinas (Ansible e Puppet).
No caso do Terraform, o processo ocorre de forma diferente, pois a partir da máquina CentOS 7, são provisionadas duas novas máquinas (uma Ubuntu 16.04 e outra Windows Server 2016) com código terraform.
A seguir, apresentaremos o processo de configuração dos arquivos de infraestrutura como código para cada uma das ferramentas, e na seção seguinte (Resultados), iremos mostrar a avaliação das ferramentas em termos das métricas escolhidas.

5.1. Ansible

Como foi mencionado na seção de fundamentação teórica, o Ansible utiliza da linguagem Yaml para escrita de arquivos de configuração de infraestrutura. Após a instalação do Ansible [5], partimos para a criação do projeto. Inicialmente, é necessário criar um arquivo de inventário, onde esse arquivo é responsável por manter organizadas as informações de acesso às máquinas hosts. Neste arquivo, que chamaremos de "inventory" (imagem abaixo), podemos definir grupos que geralmente representam os nomes dos hosts, e para esses grupos podem ser criadas seções para variáveis (sufixo ":vars") que devem armazenar as variáveis de conexão que fazem parte da conexão daquele host.

inventory

```
1  [vm_centos]
2  172.25.0.101
3
4  [vm_centos:vars]
5  ansible_connection=ssh
6  ansible_ssh_user=root
7  ansible_ssh_pass=q1w2e3R$
8
9  [vm_windows]
10 172.25.0.106
11
12 [vm_windows:vars]
13 ansible_user=administrator
14 ansible_password=q1w2e3R$
15 ansible_connection=winrm
16 ansible_port=5986
17 ansible_winrm_transport=basic
18 ansible_winrm_server_cert_validation=ignore
19
```

Trecho de código 1: Arquivo de inventário Ansible.

5.1.1. Configuração de máquina virtual CentOS 7

Em nosso playbook que será executado para a máquina virtual CentOS 7, iremos instalar o Apache Server e iniciar o serviço Apache na máquina. O playbook está descrito abaixo.

centos_tasks.yml

```
---
- hosts: vm_centos
  remote_user: root
  become: yes
  gather_facts: no

  tasks:
  - name: Apache latest version installation
    yum:
      name: httpd
      state: latest

  - name: Enable service to start on boot up
    service:
      name: httpd
      state: started
    notify: Restart apache service

  handlers:
  - name: Restart apache service
    service:
      name: httpd
      state: restarted
```

Trecho de código 2: Playbook ansible para configuração da máquina virtual CentOS 7.

Após a execução desse playbook com o comando "*ansible-playbook -i inventory centos_tasks.yml*", a página teste do servidor HTTP Apache estará disponível em: *http://localhost:80*.

5.1.2. Configuração de máquina virtual Windows 10

Para a instalação do Apache em uma máquina virtual Windows 10 com o Ansible, utilizamos o chocolatey, que é um gerenciador de pacotes para Windows. Diferentemente do CentOS 7, não foi necessário habilitar o serviço, pois após a instalação do apache, o serviço já é habilitado automaticamente.

windows_tasks.yml

```
---
- hosts: vm_windows
  tasks:

    - name: Install Apache
      win_chocolatey:
        name: apache-httpd
        state: latest
```

Trecho de código 3: Playbook ansible para configuração da máquina virtual Windows 10.

Após a execução desse playbook com o comando "*ansible-playbook -i inventory windows_tasks.yml*", a página teste do servidor HTTP Apache estará disponível através em: *http://localhost:8080*.

5.2. Puppet

O gerenciamento de configuração no Puppet é feito através de Puppet Master e Puppet Agents, onde o Puppet Master é responsável por gerenciar o/os nó(s) (Puppet Agents) e armazenar as configurações de infraestrutura (código) que devem ser aplicadas em cada um das máquinas nós.
O processo de gerenciamento da infraestrutura é feito da seguinte forma: 1 - O agente (Puppet agent) inicia a comunicação com o servidor (Puppet server), fornecendo detalhes do host e solicita o catálogo de configuração; 2 - Com base nas informações recebidas pelo agente, e na configuração definida pelo administrador, o servidor compila um catálogo e envia ao agente; 3 - O agente recebe o catálogo e aplica as configurações no host; 4 - Ao final da execução, o agente envia um relatório ao server do que foi executado/alterado no host.
Desse modo, em nosso estudo utilizamos três máquinas (sendo duas CentOS 7 e uma Windows 10), uma CentOS 7 que desempenha a função de Puppet Server e as outras duas de Puppet Agent. E no arquivo de configuração que fica na máquina Puppet Master, definiremos um código de instalação e iniciação do serviço do Apache tanto para CentOS 7, como para Windows Server.

5.2.1. Configuração de máquina virtual CentOS 7

Primeiramente, é necessário fazer a instalação do Puppet server na máquina que será responsável por gerenciar as máquinas nós, e instalar o Puppet agent em cada uma das máquinas nós [6] (no nosso caso, instalamos em apenas uma).
Para a configuração do Puppet Master, primeiramente, precisamos editar o arquivo */etc/hosts* e criar o mapeamento entre os ip 's das máquinas (Agent e Server) e um nome de domínio que será usado para identificar cada uma das máquinas. Logo em seguida, é necessário editar o arquivo puppet.conf que está localizado em */etc/puppetlabs/puppet/puppet.conf*, para definir nomes de dns alternativos, nome de domínio para a máquina Puppet Server, ambiente de execução da máquina. Após isso, iniciamos o serviço do Puppet Server com o systemctl.
Para a configuração do Puppet Agent, primeiro é necessário instalá-lo na máquina que será a agente. Em seguida, faremos o mesmo processo de editar o arquivo */etc/puppetlabs/puppet/puppet.conf*, para definir os mesmos dados que foram definidos para o Puppet Server, alterando apenas os nomes de domínio para a máquina agente.

Após essa configuração básica, executamos o comando */opt/puppetlabs/bin/puppet resource service puppet ensure=running enable=true* para executar o Puppet Agent no servidor, e nesse momento, o mesmo está tentando se conectar ao Puppet Master. Em seguida, voltando ao Puppet Master, precisamos listar a assinatura de certificado pendente (CSR) que foi enviada pelo Puppet Agent e assinar o certificado. Desse modo a configuração está concluída.

Agora partimos para a configuração do Apache na máquina Agent. Para isso, iremos até a pasta */etc/puppetlabs/code/* e criaremos um manifesto do puppet chamado *apache-install-centos7.pp,* nele adicionaremos um código puppet para realizar a instalação do serviço do Apache e colocar o serviço em execução. Por fim, da máquina Agent, executamos o comando */opt/puppetlabs/bin/puppet agent --test* para que a máquina recupere a última versão do manifesto que foi registrado na máquina Master, e aplicá-lo no Agent. Após a realização desse processo, a página padrão de instalação do apache será exibida em: *http://localhost:80*.

```
puppet_project > apache-install-centos7.pp > ...
1   node 'puppet1.domain.com.br' {
2       package { 'httpd':
3           ensure  => 'installed',
4       }
5       service { 'httpd':
6           ensure => running,
7           enable => true
8       }
9   }
10
```

Trecho de código 4: arquivo puppet para configuração do apache em vm CentOS 7.

5.2.2. Configuração de máquina virtual Windows 10

De modo análogo ao que foi feito na subseção anterior, utilizamos a mesma máquina Master CentOS 7 que foi utilizada para a configuração do Agent CentOS 7. Porém nessa seção, utilizamos uma máquina Windows 10 como Puppet Agent [8], e após configurada, ela irá obter a configuração da máquina Master e aplicar em seu nó.

Primeiramente, é necessário localizar o arquivo de hosts no Windows para mapear os endereços IP e configurar um nome de domínio da máquina server e da máquina agente, este arquivo no Windows fica localizado em: *c:\Windows\System32\Drivers\etc\hosts.*

Em seguida, na máquina agente, precisamos editar o arquivo */etc/puppetlabs/puppet/puppet.conf*, adicionando as chaves *certname* e *server*, que representam, respectivamente, o domínio que foi definido para o agente (puppet2.domain.com.br) e o domínio da máquina master (puppetserver.domain.com.br). Além do atributo *environment* que define o tipo de ambiente em que serão executados esses códigos, desse modo, o puppet permite a segregação de ambientes. Também é necessário adicionar uma regra de firewall, permitindo que o puppet master possa acessar a porta 8140 (porta padrão do puppet) via ssh.

Após esse processo, precisamos instalar um módulo Puppet, que é o *apache [9],* esse módulo instala, configura e gerencia o Apache e seus hosts virtuais, serviços web e módulos. Instalamos o módulo com o seguinte comando: *puppet module install puppetlabs-apache --version 7.0.0.* Com o módulo instalado, iremos criar um arquivo de manifest com a instalação básica do apache no windows e iniciação do serviço. Para isso, colocamos o seguinte trecho de código no arquivo */etc/puppetlabs/code/environments/production/modules/apache/manifests/apache_install_windows.pp.*

```
apache-install-windows.pp U
puppet_project > apache-install-windows.pp > class: 'apache'
1   class { 'apache': }
```

Trecho de código 5: arquivo puppet para configuração básica do apache no Windows 10.

Após a definição do manifesto puppet na máquina Master, seguimos com o mesmo processo que foi feito na subseção anterior: 1 - executar o comando */opt/puppetlabs/bin/puppet resource service puppet ensure=running enable=true* para executar o Puppet agent no servidor e abrir uma solicitação de conexão ao Puppet master. Logo após, voltamos ao Puppet Master e listamos os certificados com o comando */opt/puppetlabs/bin/puppet cert list*, e aparecerá a assinatura de certificado pendente (CSR) com o nome de domínio da máquina que solicitou (no nosso caso, a puppet agent), então com o comando *puppet cert sign puppet1.domain.com.br.* Após isso, o Puppet agent estará devidamente conectado ao Puppet master e resta apenas aplicar a configuração que está no master no agente.

Por fim, da máquina agente, para obter a configuração da máquina master e aplicá-la, basta executar o comando */opt/puppetlabs/bin/puppet agent --test.* Ao terminar a execução do comando, o Apache terá sua instalação básica funcionando perfeitamente na máquina agente Windows 10, exibindo a página de teste do Apache em: *http://localhost:8080*.

5.3. Terraform

O Terraform é uma ferramenta open-source IaC que fornece um fluxo de trabalho e CLI (interface de linha de comando) consistente para gerenciar centenas de serviços de cloud.

Para os testes dessa ferramenta, optamos por utilizar a plataforma de cloud Microsoft Azure, com uma conta gratuita com crédito promocional para utilização de

serviços de cloud. Além de ser necessária a instalação do Terraform [7] na máquina que irá realizar as configurações.

5.3.1. Provisionamento e configuração de máquina virtual Ubuntu 16.04

O Terraform possui um grande diferencial em relação às outras ferramentas de automação de infraestrutura, pois ela tem a capacidade de não apenas configurar máquinas virtuais, mas também de provisionar máquinas virtuais e customizar toda a configuração do sistema.
Nesse estudo, iremos provisionar uma máquina virtual Linux Ubuntu em sua versão 16.04. Após o provisionamento, será feita a atualização do repositório apt-get, instalação do Apache e inicialização do serviço.
No provisionamento de máquinas virtuais com o Terraform, utilizamos o provider **azurerm**, o Azure Provider pode ser usado para configurar infraestrutura no Microsoft Azure usando a api do Azure Resource Manager. Com a utilização desse provider, também se faz necessária a configuração de alguns recursos para o provisionamento da máquina virtual, entre estes estão: Grupo de recursos, Rede privada, sub rede, IP público, interface de rede, e por fim, a própria máquina virtual.
A sequência de comandos utilizados foi: *terraform init* -> para baixar as dependências dos providers que foram incluídas no arquivo principal de configuração do terraform. *terraform plan* -> o comando basicamente cria um resumo da configuração que será aplicada, de acordo com o que foi definido em seu arquivo de configuração. obs: erros de sintaxe serão acusados, caso existam. E por fim, *terraform apply* -> aplica as configurações que foram exibidas no resumo do "terraform plan" (se não houverem erros de sintaxe) e cria o ambiente de infraestrutura que foi definido no arquivo principal de configuração do terraform.
Na configuração fizemos a atualização dos pacotes existentes na máquina, alterações no firewall, instalação do Apache e a inicialização do serviço do Apache.
Ao fim da execução do comando *terraform apply,* a página padrão do Apache estará disponível em *http://localhost:80*.

```
terraform_linux_project > main.tf > ...
}

# connects the newly created virtual machine and does apt-get update,
## apache installation, firewall tweaks to allow access to apache and
### start apache service.
resource "azurerm_virtual_machine_extension" "terraformvm" {
  name                 = "vm01"
  virtual_machine_id = azurerm_virtual_machine.vm.id
  publisher            = "Microsoft.Azure.Extensions"
  type                 = "CustomScript"
  type_handler_version = "2.0"

  settings = <<SETTINGS
    {
      "commandToExecute": "apt-get update -y && \
        apt-get install apache2 -y && \
        ufw app list && \
        ufw allow 'Apache Full' && \
        ufw status && \
        systemctl status apache2"
    }
  SETTINGS
}
```

Trecho de código 6: Código Terraform para configuração da máquina Ubuntu recentemente criada.

O código completo que foi desenvolvido para provisionamento e configuração da máquina virtual Ubuntu pode ser acessado neste link.
Sequência de comandos: terraform init, terraform plan, terraform apply.

5.3.2. Provisionamento de máquina virtual Windows Server 2016

De modo análogo ao que foi feito para a máquina virtual Ubuntu 16.04, fizemos o provisionamento de uma máquina virtual Windows Server 2016 a partir de código Terraform. A instalação do Apache não foi possível de fazer, pois ela não é feita através de somente códigos PowerShell.
O código desenvolvido para provisionamento da máquina virtual Windows Server pode ser acessado neste link.

6. RESULTADOS

Há algumas diferenças de modo geral entre as ferramentas Ansible, Puppet e Terraform. Algumas nem tão grandes e outras que diferem bastante uma da outra. Logo abaixo, temos considerações mais específicas sobre cada uma das ferramentas para cada uma das métricas de qualidade que escolhemos para compor o nosso estudo.

6.1. Manutenibilidade

Todos os comentários descritos nesta seção são baseados em minhas experiências com as ferramentas durante esse período de estudos em que testei as ferramentas, dentro de minhas limitações, a fim de encontrar vantagens e desvantagens para cada uma das ferramentas, e com base nos resultados, definir qual ferramenta se sai melhor em determinado contexto.
No que diz respeito a métrica de Manutenibilidade, atráves das poucas experiências que tive com cada uma das ferramentas, a ferramenta mais simples de se realizar manutenções, no que diz respeito ao código, foi a ferramenta Ansible. Já a ferramenta Terraform, não é

complexa de realizar manutenções, porém devemos prestar atenção ao configurar providers e resources específicos, pois suas configurações devem ser seguidas conforme é especificado na documentação de cada um dos módulos que são utilizados. Por fim, em relação ao Puppet, a ferramenta, na minha opinião se apresentou como bastante complexa, desde sua configuração, que inclui a configuração tanto da máquina agente, como da máquina master, além dos scripts precisarem ficar em uma pasta específica do puppet (a partir do que foi estudado da ferramenta, não foi encontrada uma forma diferente de criar manifestos de configuração).

6.2. Portabilidade

6.2.1. Ansible

A ferramenta Ansible foi bastante simples em sua utilização, principalmente em máquinas Linux, onde foi feita a configuração de máquinas CentOS 7 e Windows 10 utilizando como máquina master uma máquina Ubuntu para disparar as configurações. Entretanto, para utilização de uma máquina Windows 10 para disparar as configurações para as máquinas nós, se tornou bem mais problemático, pois tínhamos que utilizar um ambiente virtualizado que fornece funcionalidades semelhantes a uma distribuição Linux no Windows (ex: Cygwin [11]), instalar o Ansible [12] e assim, poder executar comandos Ansible no Windows. Por este motivo, não realizamos os testes disparando as configurações de uma máquina Windows, além da limitação de tempo disponível para a conclusão do trabalho.
Outra impressão que tive com o Ansible, é que ela melhor se adapta ao Linux, pois uma máquina bastion (mestre) de distribuição Linux pode configurar com facilidade máquinas Linux e Windows. Já uma máquina bastion Windows, é necessária configurações adicionais e que de certa forma, dependem de um virtualizador Linux para realizar a mesma tarefa.

6.2.2. Puppet

A ferramenta de automação Puppet foi a mais difícil de ser configurada. Inúmeros problemas ocorreram na instalação do Puppet Master e Puppet Agent. Inicialmente foram feitos testes para a instalação no Ubuntu mas diversos problemas ocorreram e, devido a isso, foi realizada a instalação em uma máquina com Sistema Operacional CentOS 7, que possui maior suporte à ferramenta e com isso a instalação ocorreu sem problemas.
Outra impressão com a experiência com o Puppet, é que a curva de aprendizado é mais inclinada, pois ele possui sua linguagem própria de codificação. Devido a isso, se faz necessário consultar sua documentação e os exemplos apresentados não são intuitivos, o que aumenta a complexidade em seu entendimento.

6.2.3. Terraform

O Terraform é uma ferramenta bastante completa, pois permite provisionar e realizar configurações em máquinas virtuais, essencialmente em ambientes cloud (Microsoft Azure, AWS, Google Cloud, etc.). Em relação a portabilidade, o Terraform funcionou muito bem, tanto para o provisionamento da máquina Ubuntu, como para o provisionamento da máquina Windows Server. O que na minha opinião, é suficiente para que a ferramenta Terraform tenha nota máxima no quesito portabilidade.

6.3 Usabilidade

6.3.1. Ansible

Mesmo sendo uma ferramenta CLI (interface de linha de comando), o Ansible apresenta diversos pontos que beneficiam a usabilidade do sistema e tornam o aprendizado bastante simplificado, a usabilidade é medida pela eficácia, eficiência e satisfação na realização de uma tarefa utilizando a ferramenta.
Um ponto importante é a curva de aprendizado da ferramenta que não é acentuada, pois com poucas horas de estudo da ferramenta, já é possível criar suas próprias automações.
Os comandos ad-hoc que a ferramenta utiliza são bastante intuitivos, além da documentação e dos vastos materiais que são encontrados na internet, com os mais diversos exemplos.

6.3.2. Puppet

A ferramenta Puppet em termos de usabilidade, na minha opinião, é inferior a usabilidade da ferramenta Ansible, pois o processo de configuração do ambiente para uma simples automação é demasiadamente complexo. Outro ponto importante que deixa o processo de configuração mais complexo é o fato de ser necessário instalar o Puppet tanto na máquina Master como também em todas as máquinas Agentes.

6.3.3. Terraform

A usabilidade do Terraform, de forma análoga ao Ansible, é simples e intuitiva, pois mesmo sendo uma ferramenta CLI, os comandos são bem intuitivos, a instalação e codificação da ferramenta também são simples, bastando seguir as recomendações de utilização e configuração dos providers e resources que a próprio site oficial da ferramenta oferece.

7. CONCLUSÃO

Em nosso estudo, realizamos pesquisas comparando diversas ferramentas de infraestrutura como código. Foi observado que as ferramentas diferem umas das outras

em termos de complexidade, curva de aprendizado, capacidades e limitações.

Em relação a diferenciais, a ferramenta Terraform saiu à frente das demais, pois com ela é possível não somente configurar uma máquina virtual, mas também provisioná-la. No entanto, não foi possível configurar o Apache na máquina Windows, pois o processo de instalação do Apache no Windows não se dá através da execução de um simples comando Powershell, além das limitações de tempo que tivemos para a realização da pesquisa e criação do presente documento.

A ferramenta Puppet se apresentou como bastante complexa, principalmente na instalação da ferramenta nas máquinas agentes e na máquina master, devido a arquitetura do Puppet ser agent-to-master. O fato de ser necessária a instalação do Puppet tanto nas máquinas agentes e master, torna o processo bastante complexo.

Por outro lado, a ferramenta Ansible é agentless (livre de agentes), ou seja, não é necessária a execução de nenhuma configuração manual nas máquinas que serão gerenciadas, isso, na minha opinião, já torna o Ansible superior ao Puppet.

Na minha opinião, o uso combinado de Ansible + Terraform pode garantir qualquer configuração de infraestrutura, pois com o Terraform iríamos provisionar a máquina virtual, seja ela de qualquer sistema operacional e com o Ansible iríamos nos conectar na máquina recentemente criada e executar as configurações desejadas.

Uma possibilidade de continuação e aprimoramento deste trabalho, seria a utilização das ferramentas para a configuração de várias máquinas, e não apenas uma (como foi utilizada neste trabalho). Pois, com isso poderíamos avaliar outras métricas de qualidade de software, como por exemplo a eficiência e velocidade de conexão e configuração.

Por fim, esse estudo foi realizado dentro das limitações de tempo que tivemos pelo curto espaço de tempo do período letivo. Todas as opiniões que foram dadas neste estudo foram embasadas nas vivências que tive com cada uma das ferramentas e as conclusões de cada uma das ferramentas podem não ser diferentes das minhas, levando em consideração que o estudo seja realizado por outra pessoa.

8. AGRADECIMENTOS

Gostaria de agradecer primeiramente a Deus, por me abençoar com saúde e me permitir chegar a conclusão de mais uma etapa da minha vida. Logo em seguida, agradeço a minha mãe e a minha avó (in memoriam), por me incentivarem a nunca desistir, e que sempre me apoiaram e estiveram ao meu lado durante a graduação, e principalmente, nos momentos mais difíceis. Sem sombra de dúvidas, essa conquista dedico às mulheres da minha vida e ao meu pai (in memoriam).

Agradeço a todos os amigos que fizeram parte da minha graduação e tiveram grande importância em meu crescimento pessoal e profissional, especialmente a Abel, Hércules e Samuel. Agradeço também, a todos os amigos que trabalharam comigo, sejam em projetos ou trabalhos em grupo, e por todas as discussões de valor inestimável que tivemos sobre os mais diversos assuntos.

Agradeço a Fubica (Francisco Brasileiro) e João Arthur, por me proporcionarem a minha primeira experiência em projetos. Isso me permitiu desenvolver minhas habilidades técnicas e aprender muito com as suas experiências.

Agradeço a Reinaldo por todo o acompanhamento, suporte e apoio enquanto mentor deste trabalho, assim como suporte e trabalho desempenhados durante as aulas das disciplinas de Redes e Administração de Sistemas.

Por fim, agradeço a todos os amigos e colegas que conquistei durante a graduação, sendo todos de grande importância para que eu chegasse até aqui.

9. REFERÊNCIAS

[1] Systems and software Quality Requirements and Evaluation (SQuaRE), em: https://www.iso.org/obp/ui/#:~:text=4.2%C2%A0%C2%A0%C2%A0Product%20quality%20model

[2] How Ansible works?, em: https://www.ansible.com/overview/how-ansible-works?hsLang=en-us

[3] Introduction to Puppet, em: https://puppet.com/docs/puppet/6/puppet_overview.html

[4] Terraform Overview, em: https://www.terraform.io/

[5] Ansible: Instalação e Configuração, em: https://www.zentica-global.com/pt/zentica-blog/ver/como-instalar-o-ansible-no-ubuntu-20.04-607398fe782b3

[6] Install Puppet on CentOS7 machine, em: https://bobcares.com/blog/install-puppet-on-centos-7/

[7] Install Terraform, em: https://learn.hashicorp.com/tutorials/terraform/install-cli

[8] Install Puppet Agent on Windows, em: https://puppet.com/docs/puppet/7/install_agents.html#install_windows_agents

[9] Apache module on Puppet, em: https://forge.puppet.com/modules/puppetlabs/apache#beginning-with-apache

[10] Ferramentas de gerenciamento de configuração Devops, em: https://kinsta.com/pt/blog/ferramentas-devops/

[11] This is the home of the Cygwin project, em: https://www.cygwin.com/

[12] How to install and configure Ansible on Windows, em:
https://phoenixnap.com/kb/install-ansible-on-windows

Sobre o Autor:

Paulo Mendes da Silva Júnior, graduando em Ciência da Computação, há cerca de 6 anos começou sua jornada no mundo de TI cursando técnico em tecnologias da informação, logo após, ingressou na graduação, e ao longo do tempo, buscou se especializar nas áreas de Engenharia de Software, Desenvolvimento Web e Infraestrutura em TI. Participou por 1 ano como FullStack Developer no projeto Atmosphere no Laboratório de Sistemas Distribuídos - LSD, além de ter atuado como estagiário em Qualidade de Software na empresa Vsoft Tecnologia. Atualmente atua como Engenheiro de Software na empresa F9C Security.