Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Bacharelado em Sistemas de Informação

Desafios na Gestão de Infraestrutura de TI em Ambientes Híbridos: Uma Abordagem Experimental

Thomas Batista dos Santos

JUIZ DE FORA
JULHO, 2024

Desafios na Gestão de Infraestrutura de TI em Ambientes Híbridos: Uma Abordagem Experimental

Thomas Batista dos Santos

Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Bacharelado em Sistemas de Informação

Orientador: Mario Antônio Ribeiro Dantas

JUIZ DE FORA
JULHO, 2024

Desafios na Gestão de Infraestrutura de TI em Ambientes Híbridos: Uma Abordagem Experimental

Thomas Batista dos Santos

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM SISTEMAS DE INFORMAÇÃO.

Aprovada por:

Mario Antônio Ribeiro Dantas
Doutor

Luciano Jerez Chaves
Doutor

Victor Stroele de Andrade Menezes
Doutor

JUIZ DE FORA
5 DE JULHO, 2024

A quem sempre esteve ao meu lado
durante essa longa jornada.

Resumo

A evolução da computação em nuvem e a adoção de técnicas como a infraestrutura como código (IAC), têm transformado a maneira como empresas lidam com seus recursos de TI. Essas tecnologias são fundamentais para atualizar a infraestrutura, possibilitando maior flexibilidade, capacidade de expansão e eficiência operacional. O propósito deste trabalho é oferecer uma perspectiva experimental sobre a implementação da infraestrutura como código em um ambiente corporativo real, com o objetivo de efetuar um processo de modernização e automatização da gestão e configuração da infraestrutura através da implementação das ferramentas Ansible e vRx. Os resultados experimentais indicaram que a adoção de infraestrutura como código proveu um diferencial na gerência de infraestrutura e reduziu o esforço gasto no dia a dia da corporação.

Palavras-chave: Ansible, Nuvem, Infraestrutura como Código, Infraestrutura, vRx.

Abstract

The evolution of cloud computing and the adoption of techniques such as infrastructure as code (IAC) have transformed the way companies deal with their IT resources. These technologies are essential for updating infrastructure, enabling greater flexibility, expansion capacity and operational efficiency. The purpose of this work is to offer an experimental perspective on the implementation of infrastructure as code in a real corporate environment, with the aim of carrying out a process of modernization and automation of infrastructure management and configuration through the implementation of Ansible and vRx tools. The experimental results indicated that the adoption of infrastructure as code provided a difference in infrastructure management and reduced the effort spent in the corporation's day-to-day operations.

Keywords: Ansible, Cloud, Infrastructure as code, Infrastructure, vRx.

Agradecimentos

A todos os meus parentes e amigos, pelo encorajamento e apoio.

Ao professor Mario Dantas pela orientação, amizade e principalmente, pela paciência, sem a qual este trabalho não se realizaria.

Aos professores do Departamento de Ciência da Computação pelos seus ensinamentos e aos funcionários da UFJF, que durante esses anos, contribuíram de algum modo para o nosso enriquecimento pessoal e profissional.

A empresa que apoiou a realização desse projeto, provendo a infraestrutura e os desafios.

"Eu ainda estou de pé depois de todo esse tempo, parecendo um verdadeiro sobrevivente, sentindo-me como uma criança".

Elton John (I'm Still Standing)

Conteúdo

Lista de Figuras

Lista de Abreviações

AWS	Amazon Web Services
CMDB	*Configuration Management Database*
CVE	*Common Vulnerabilities and Exposures*
DCC	Departamento de Ciência da Computação
UFJF	Universidade Federal de Juiz de Fora
EOL	*End of Life*
IAC	Infraestrutura como código
MFA	*Multifactor Authentication*
NIST	Instituto Nacional de Padrões e Tecnologia dos EUA
PAYG	*Pay as you Go*
RDP	Protocolo de Área de Trabalho Remota
SLA	*Service Level Agreement*
SSH	*Secure Shell*
SSO	*Single Sign-On*
ITSM	*IT Service Management*
WAF	*Well-Architected Framework*

1 Introdução

1.1 Apresentação do tema

Nos últimos anos a computação em nuvem e a adoção de práticas de infraestrutura como código (IAC) têm revolucionado a forma como as organizações gerenciam e provisionam recursos de TI. Essas tecnologias têm se mostrado essenciais para a modernização da infraestrutura, permitindo maior agilidade e eficiência operacional em empresas, como observado em (SOUZA; FRANCO; SILVA, 2023).

Uma vez em que o paradigma da tradicional infraestrutura de TI deixou de ser necessariamente a única opção a ser seguida, novos paradigmas como Computação em Nuvem e Nuvem Híbrida, que promovem elasticidade e uso sob demanda, começaram a ganhar destaque (MELL P.; GRANCE, 2011). No entanto, essa mudança de paradigma também apresenta novos desafios significativos, tornando importante seguir boas práticas de adoção a nuvem, garantindo a segurança do ambiente. A mudança de infraestrutura também pode ter impactos significativos no modelo de *data centers*, incluindo a necessidade de redesenhar a arquitetura de TI e repensar a forma como a infraestrutura é gerenciada e escalada.

1.2 Motivação

Com isso em mente, o trabalho apresentado aqui surgiu da necessidade de melhoria no gerenciamento de infraestrutura em uma empresa de logística. Os funcionários da equipe de infraestrutura de TI da empresa em questão constantemente precisam realizar novas configurações em seus ativos, e também criar novos recursos conforme demanda. Dados o tamanho e a complexidade do ambiente, a execução dessas tarefas requer atenção recorrente da equipe, e a falta de procedimentos, ferramentas e técnicas mais avançadas acabam resultando em diversas atividades manuais que consomem muitas horas ao longo da semana. O trabalho aqui desenvolvido é um experimento de melhoria de gestão e

espera-se que seus resultados sejam positivos, sirvam de orientação e abra portas para novas melhorias na equipe.

1.3 Pergunta de Pesquisa

Com o crescimento do mercado em nuvem hoje temos várias opções de infraestrutura, porém cada uma possui diferentes características, vantagens e desvantagens que acabam modificando o seu gerenciamento. É importante que esse gerenciamento seja feito de forma segura, ágil e padronizada. Tendo isso em mente, a pergunta de pesquisa é: "Quais são os principais desafios na modernização do gerenciamento da infraestrutura de TI?"

1.4 Objetivo Geral

Analisar os possíveis desafios enfrentados na modernização de infraestrutura de TI e realizar uma abordagem experimental em um ambiente corporativo.

1.5 Objetivos Específicos

- Realizar uma revisão de literatura sobre o tema;
- Apresentar uma proposta de abordagem para a modernização de infraestrutura de TI;
- Efetuar uma abordagem experimental utilizando infraestrutura como código;
- Demonstrar resultados dos esforços da abordagem;
- Propor novas abordagens complementares.

1.6 Estrutura do Trabalho

O primeiro capítulo do trabalho tem como objetivo introduzir o tema, abordando a motivação, definindo a pergunta de pesquisa e estabelecendo os objetivos gerais e específicos. No segundo capítulo, serão apresentados conceitos importantes e relevantes, fornecendo

uma visão geral e criando uma base sólida para o desenvolvimento do projeto. O terceiro capítulo revisa alguns trabalhos relacionados ao tema, que serviram como fonte de informação ou inspiração para a realização deste estudo. No quarto capítulo, é apresentada uma proposta de utilização de IAC em um ambiente corporativo e seus resultados. Finalmente, no quinto capítulo, serão apresentadas as conclusões e possíveis trabalhos futuros.

2 Fundamentação Teórica

2.1 Introdução

Nesse capítulo serão apresentados conceitos fundamentais relacionados ao tema do trabalho como infraestrutura, modelos de implantação e serviços em nuvem e infraestrutura como código.

2.2 Infraestrutura On-premises

A Infraestrutura *On-premises*, também conhecido como nuvem privada, é o modelo tradicional de implantação e gerenciamento de recursos de TI. Nesse modelo, a organização mantém todos os *hardwares*, *softwares* e equipamentos de rede em um ou mais *data centers* físicos que lhe pertencem, fazendo assim com que toda a responsabilidade e gerência deles seja feita de forma interna ou terceirizada (MELL P.; GRANCE, 2011).

Por ter seus recursos de forma local e ser responsável pelo seu controle, as organizações são capazes de ter um nível de customização do ambiente mais elevado, podendo assim adaptar seu ambiente conforme necessário, definindo medidas de segurança para garantir que seus ativos sejam manuseados de forma correta, segura e somente por pessoas capacitadas. A performance do ambiente é consistente e possui uma latência baixa devido a proximidade dos recursos, porém é necessário uma equipe responsável pela manutenção do ambiente, realizando atualizações de segurança nos ativos, *softwares* e inspeção local dos equipamentos.

O modelo *On-premises* traz uma certa dificuldade de escalonamento, pois modificar o ambiente está atrelado a compra e investimento em configurações de servidores, *softwares*, e outros custos de infraestrutura (FISHER, 2018). Caso o ambiente seja dinâmico, isso pode impactar na implementação de novos projetos, resultando em atrasos ou gastos não previstos. Esse é um problema que pode ser resolvido na utilização do modelo de Computação em Nuvem, que será apresentado na próxima sessão.

2.3 Computação em nuvem

O NIST define Computação em nuvem como "Um modelo que permite acesso onipresente, conveniente e sob demanda à rede para um ambiente compartilhado com um conjunto de recursos de computação configuráveis, como por exemplo, redes, servidores, armazenamento, aplicativos e serviços, que podem ser rapidamente provisionados e liberados com mínimo esforço de gerenciamento ou interação com o provedor de serviços."(MELL P.; GRANCE, 2011). Tanto a computação em nuvem quanto outros paradigmas que surgiram no início dos anos 2000 ganharam força ao longo dos anos, gerando um crescimento de publicações científicas como mostrado pela Figura 2.1, retirada de (DONNO; TANGE; DRAGONI, 2019).

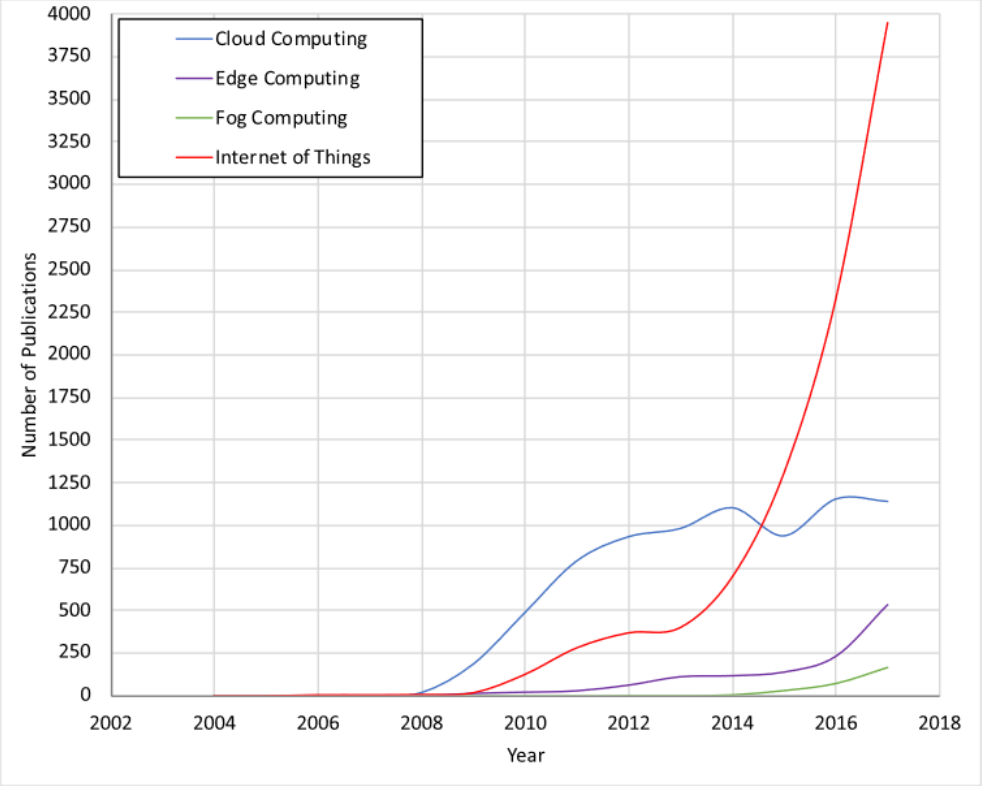

Figura 2.1: Artigos relacionados a nuvem e Internet das Coisas (IoT) entre 2002 e 2018.

2.3.1 Características

Segundo (MELL P.; GRANCE, 2011), podemos dizer que algumas características da Computação em nuvem são:

- **Autoatendimento sob demanda:**

Um consumidor pode fornecer unilateralmente recursos de computação, como tempo do servidor e armazenamento de rede, conforme necessário, automaticamente, sem a necessidade de humanos interagindo com cada provedor de serviços.

- **Amplo acesso à rede:**

  Os recursos estão disponíveis na rede e são acessados por meio de mecanismos padrão que promovem o uso por plataformas heterogêneas de clientes finos ou grossos (por exemplo, *smartphones, tablets, laptops* e estações de trabalho).

- **Agrupamento de recursos:**

  Os recursos de computação do provedor são agrupados para atender vários consumidores usando um modelo multi-locatário, com diferentes recursos físicos e virtuais atribuídos e reatribuídos dinamicamente de acordo com a demanda do consumidor. Há uma sensação de independência de localização, pois o cliente geralmente não tem controle ou conhecimento sobre a localização exata dos recursos fornecidos, mas pode especificar a localização em um nível mais alto de abstração (por exemplo, país, estado ou *data centers*). Exemplos de recursos incluem armazenamento, processamento, memória e largura de banda de rede.

- **Elasticidade rápida:**

  As capacidades podem ser provisionadas e liberadas de forma elástica, em alguns casos automaticamente, para escalar rapidamente para fora *(scale-out)* e para dentro *(scale-in)*, proporcionalmente à demanda. Para o consumidor, as capacidades disponíveis para provisionamento muitas vezes parecem ilimitadas e podem ser apropriado em qualquer quantidade a qualquer momento.

- **Serviço medido:**

  Os sistemas em nuvem controlam e otimizam automaticamente o uso de recursos, aproveitando uma capacidade de medição em algum nível de abstração apropriado ao tipo de serviço (por exemplo, armazenamento, processamento, largura de banda e contas de usuários ativas). O uso de recursos pode ser monitorado, controlado e

relatado, proporcionando transparência tanto para o provedor quanto para o consumidor do serviço utilizado.

2.4 Modelos de Implantação na nuvem

2.4.1 Nuvem Pública

A nuvem pública é provisionada para uso do público em geral. Fica localizada nas instalações do provedor de nuvem (Ex.:Microsoft, Google, Amazon ou Oracle) e pode ser utilizado por organizações acadêmicas, governamentais e empresas. Apesar de ter como padrão o modelo *Pay as You Go* (PAYG), onde a cobrança vem a medida do uso, também é possível fazer reservas de diversos recursos para baratear os custos.

2.4.2 Nuvem Privada

Esse tipo de infraestrutura é provisionada de forma exclusiva para uma organização, podendo ser gerenciada e operada por ela e/ou um organização terceira(MELL P.; GRANCE, 2011). A infraestrutura *On-premises* pode ser considerada uma nuvem privada de forma local, porém esse modelo também pode ser terceirizado a um provedor de nuvem privada, em um local remoto, transferindo a responsabilidade de seu gerenciamento. A terceirização da infraestrutura pode trazer conforto e menos responsabilidade para a organização, porém pode acarretar em um custo maior.

2.4.3 Nuvem Híbrida

É a composição de duas ou mais infraestruturas de nuvem distintas (privada, comunitária ou pública) que permanecem entidades únicas, mas são unidas por tecnologia padronizada ou proprietária que permite a portabilidade de dados e aplicativos (por exemplo, intermitência de nuvem para balanceamento de carga entre nuvens)(MELL P.; GRANCE, 2011).

O paradigma de nuvem híbrida é utilizado em larga escala por empresas que estão em um momento de transição do modelo tradicional para a nuvem. O planejamento para uma migração para a nuvem é um processo que envolve diversos recursos da

organização, adaptação de aplicações, modificação da infraestrutura, segurança e custos. Entre aspectos que devem ser analisados com muito cuidado, podemos citar:

- Definição dos objetivos e metas;
- Avaliação da infraestrutura e dependências;
- Escolha das plataformas de nuvem e serviços;
- Planejamento financeiro;
- Segurança e Conformidade dos dados;
- Definição das ferramentas de monitoramento;
- Backups e recuperação de desastres;
- Treinamento dos profissionais;
- Definição do plano de execução;
- Acompanhamento em tempo real pós-migração.

2.5 Modelos de serviço na nuvem

Quando tratamos de nuvem, existem alguns modelos de serviços mais utilizados cujo gerenciamento e responsabilidades se diferem, conforme apresentado na Figura 2.2 (VERGADIA, 2022). Nessa seção vamos detalhar algumas delas.

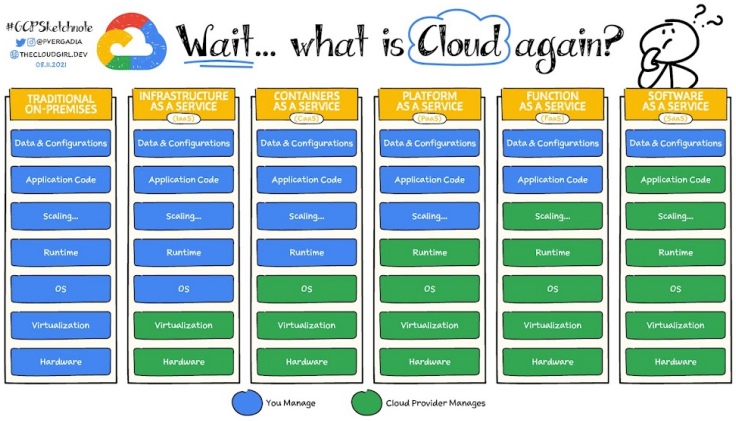

Figura 2.2: Comparação entre modelos de serviço de nuvem

2.5.1 Infraestrutura como Serviço (IaaS)

Nesse modelo de serviço, o provedor de nuvem é responsável pelo hardware e virtualização. Já o consumidor cuida dos sistemas operacionais (Ex.:Windows e Linux), armazenamento e aplicações instaladas na infraestrutura.

A IaaS é parecida com o ambiente *On-Premises* e tem como vantagem ser um pouco mais flexível, devido ao fato da organização ter controle sobre o sistema operacional, e reduz o custo de custo total de posse, pois o recurso é pago conforme o uso (ZHANG et al., 2022). Podemos citar como exemplo a criação de máquinas virtualizadas com sistemas operacionais Windows ou Linux.

2.5.2 Plataforma como Serviço (PaaS)

No modelo Plataforma como Serviço (PaaS), o provedor de nuvem é responsável pela gerência de todos os recursos de *hardware*, virtualização e *softwares* de desenvolvimento. Os consumidores da nuvem ainda precisam codificar e gerenciar dados e aplicativos, mas o ambiente para criar e implantar aplicativos é gerenciado e mantido pelo provedor de serviços em nuvem, reduzindo o tempo e esforço gasto no desenvolvimento (ZHANG et al., 2022). Podemos citar como exemplos os Banco dos Dados Azure SQL e os *Azure App Services*, gerenciados pela nuvem Microsoft Azure.

2.5.3 Função como Serviço (FaaS)

Esse é um modelo de serviço cujo provedor de nuvem é responsável pelo gerenciamento de toda infraestrutura, escalabilidade e balanceamento. O desenvolvedor (consumidor) tem como único objetivo escrever funções que são engatilhadas via modificações nos dados e eventos, podendo abstrair o provisionamento e gerenciamento de servidores onde a aplicação será executada. Podemos citar como exemplo o *Azure Functions* e o *AWS Lambda*.

2.5.4 Software como Serviço (SaaS)

Diferente do FaaS que tem como foco a execução de pequenas funções atreladas a eventos, o SaaS é um modelo que provê aplicações inteiras via internet com diversas funcionalidades. O provedor de nuvem é responsável por toda infraestrutura, como servidores, sistemas operacionais, armazenamento e redes. As manutenções e atualizações também são feitos pelo provedor, deixando ao consumidor somente a responsabilidade com as configurações específicas da aplicação (MELL P.; GRANCE, 2011). Podemos citar como exemplo os conjuntos de Softwares Microsoft 365, *PowerBI*, *Power Apps* e *Power Automate* da Microsoft.

2.6 Infraestrutura como Código (IAC)

Atualmente temos uma tendencia global de agilidade no mundo da TI, onde tarefas que antes eram feitas de forma manual agora tendem a ser automatizadas utilizando técnicas de engenharia de *software*, com devido controle de versão, pipelines e segurança com o intuito de reduzir o tempo e esforço utilizado no desenvolvimento de *softwares* e operações (BASS; WEBER; ZHU, 2015).

Podemos definir IAC como uma abordagem para automação de infraestrutura baseada em práticas de desenvolvimento de *software*. A IAC enfatiza resultados consistentes e repetíveis rotinas para provisionamento e alteração de sistemas e suas configurações. A utilização de ferramentas de automação como o Ansible, pode ser uma alternativa à configuração manual de infraestrutura, assim reduzindo a mão de obra e tempo gasto em processos automatizáveis. O versionamento do código fonte garante com que você consiga identificar falhas, produzir melhorias de forma mais fácil, e por fim executar testes e replicar as alterações em ambientes controlados (MORRIS, 2021).

2.7 Considerações do Capítulo

Nesse capítulo foram abordados os conceitos teóricos que serviram de base ao trabalho. Foram apresentados conceitos de infraestrutura de TI, partindo do modelo tradicional

On-premises, seguindo para a nuvem e suas características, modelos e serviços, e por fim apresentando a IAC. Os conceitos aqui abordados estão diretamente ligados aos trabalhos relacionados que serão apresentados no próximo capítulo e na proposta do trabalho.

3 Trabalhos Relacionados

3.1 Introdução

Nesse capítulo serão apresentados alguns trabalhos relacionados ao uso de infraestrutura como código e técnicas de DevOps em infraestruturas de TI, ilustrando suas características, desafios, melhores práticas e casos de estudo em ambientes corporativos.

3.2 Trabalhos

Em (KUMARA et al., 2021) é realizada uma análise sistemática da literatura cinzenta industrial sobre IAC, utilizando como base postagens em blogs, tutoriais e *white papers*. Utilizando técnicas qualitativas para extrair informações relevantes, o estudo visa caracterizar IAC e compilar um catálogo de melhores e piores práticas, utilizando como base três linguagens famosas de IAC: Ansible, Puppet e Chef.

A análise se da dessa forma pois segundo (KUMARA et al., 2021), ainda se tem muito pouco material acadêmico sobre o tema IAC. Empresas tem adotado a prática cada vez mais, originando uma grande quantidade de literatura cinzenta e resultando em uma grande lacuna entre os trabalhos acadêmicos com o que é praticado na indústria. Confirmando a afirmação, o estudo mostra que cerca de 67% da literatura teve como fonte blogs, 10% artigos e 9% documentações. Segundo o estudo, entre as melhores práticas a serem adotadas em IAC temos que o código deve ser escrito para pessoas e não computadores, sendo modularizado e evitando reutilização de código, é importante fazer uso das ferramentas e módulos das linguagens utilizadas, e executar mudanças incrementais e versionadas ao código, mantendo uma documentação sucinta e bem explicada. E por fim, escrever um código seguro, separando dados do código, e isolando senhas e dados sensíveis.

Em *Cloud Infrastructure Automation Through IaC (Infrastructure as Code)* (HASAN; ANSARY, 2023), dá-se uma visão geral sobre computação em nuvem, introduzindo

um pouco de seu conceito, serviços, benefícios e desafios. Como forma de contornar os desafios, apresentam o conceito de infraestrutura como código, discutem seus benefícios, desafios e fazem um paralelo entre o meio tradicional de cuidar de ambientes em nuvem com um método proposto utilizando IAC para melhorar o fluxo de trabalho e agilizar demandas repetitivas em ambientes em nuvem.

Achieving Operational Excellence in Cloud Management: Practical Evaluation of Infrastructure as Code and the Well-Architected Framework's Adoption to Improve Process Maturity (GUTTA, 2023), faz uma avaliação sobre a adoção do uso de IAC para provisionar e gerenciar recursos de nuvem. O estudo avalia o impacto da IAC na agilidade, otimização de custos e gerenciamento de riscos nas operações de nuvem e além disso, disserta sobre a aplicação da *Well-Architected Framework* (WAF), um conjunto de melhores práticas para construir infraestruturas seguras, de alto desempenho, resilientes e eficientes na nuvem. Através do estudo, o autor busca apresentar percepções pertinentes para organizações em busca de aprimoramento no gerenciamento de nuvem e discutir também os benefícios e dificuldades da implementação de IAC e adoção das melhores práticas operacionais em nuvem.

Em *Infrastructure as Code as a Foundational Technique for Increasing the DevOps Maturity Level: Two Case Studies* (SOUZA; FRANCO; SILVA, 2023), se propõe à realizar dois estudos de caso relacionados à implementação de infraestrutura como código em projetos com diferentes níveis de maturidade em DevOps. Os estudos de caso foram feitos utilizando os serviços da AWS, porém é possível também replicar o caso com outros provedores de nuvem, utilizando seus serviços correspondentes.

O primeiro projeto trata-se de uma empresa de telecomunicações que possui uma aplicação de comunicação do tipo plataforma como serviço (PaaS) na nuvem da AWS. A empresa sofria com desvios de configuração, falta de controle de versionamento e falta de padronização na implementação de versões nos ambientes produtivos e de teste. Com o intuito de resolver os problemas de configuração e implantação manual da aplicação, seguindo as melhores práticas de gerenciamento de infraestrutura e IAC, foi utilizado o AWS CloudFormation, serviço de modelagem e configuração de infraestrutura da AWS e também o Ansible, uma ferramenta *open-source* de gerenciamento, configuração e auto-

matização de servidores e aplicações.

Já o segundo projeto foi uma aplicação web utilizada por Coaches em saúde, responsáveis por dar suporte a pacientes com câncer. A aplicação tinha como restrição estar na nuvem da AWS e os desenvolvedores da empresa contratante não possuíam conhecimento prévio em gerenciamento de infraestrutura, DevOps ou IAC. Sendo assim, foi utilizado um framework Serveless e o serviço AWS Lambda para simplificar a implantação e desenvolvimento do sistema, porém mantendo as boas práticas.

Em ambos projetos foi possível identificar uma evolução na maturidade tecnológica, processos e cultura, além de agilidade na implementação e gerenciamento dos ambientes com as devidas ferramentas utilizadas.

Infrastructure as Code (QUATTROCCHI; TAMBURRI, 2023) demonstra o quanto a infraestrutura como código tem evoluído nos últimos anos, chegando ao ponto de ser considerada madura e rica em práticas focadas no desenvolvimento contínuo de softwares. O artigo contém elementos essenciais para praticantes e teóricos da IAC, e após a análise de todos os artigos avaliados chegou a conclusão de que:

- **A descentralização é a porta e o código de infraestrutura é sua chave:**

  É necessária uma gestão mais refinada da descentralização em ambientes DevOps, buscando encontrar melhores configurações de arquitetura e continuidade para infraestruturas.

- **Automação do ciclo de vida requer otimização inteligente:**

  O ciclo de vida da automação precisa ser mais proativo do que reativo através de automações inteligentes capazes de se adaptar rapidamente.

- **Modelos de maturidade para DevOps são essenciais:**

  Todos os artigos reiteram a necessidade de uma governança estruturada e inteligente na adoção do DevOps e uso de IAC.

- **Segurança é primordial nas operações:**

  É de suma importância ter fundamentos de segurança do gerenciamento de qualidade de código de infraestrutura, operações de ciclo de vida, manutenção e evolução.

3.3 Considerações do Capítulo

Nesse capítulo foram apresentados trabalhos relacionados a pesquisa, que envolvem o uso de IAC para gerenciamento e configuração de ambientes. Apresentamos alguns estudos focados em identificar as melhores práticas do uso de IAC seguindo os pilares definidos pela WAF, discutir seus conceitos, benefícios e desafios de implementação e por fim casos de uso em ambientes corporativos com o intuito de agilizar e simplificar o gerenciamento e configuração de infraestrutura. Esse trabalho visa aplicar as melhores práticas apresentadas nos trabalhos relacionados, tentando usufruir dos benefícios da IAC em um ambiente corporativo com baixa maturidade em automação e IAC, dando o pontapé inicial em uma sequência de ações focadas na melhoria na gestão de uma nuvem híbrida.

4 Proposta

4.1 Introdução

Nesse capítulo será apresentada uma proposta experimental de utilização de ferramentas de automação via infraestrutura como código para gerenciar uma infraestrutura de larga escala em um ambiente corporativo. Essas ferramentas serão utilizadas também para a implantação em larga escala do Zabbix na sua versão 6.0, uma ferramenta *open-source* de monitoramento de infraestrutura de TI.

4.2 Ambiente atual

O ambiente a ser explorado é a infraestrutura de TI de uma empresa de logística, cujo parque possui mais de 500 servidores, sendo eles físicos e virtualizados *On-premises* e na nuvem, com diferentes sistemas operacionais, espalhados por diversas redes internas com devidas ferramentas de segurança monitorando e gerenciando o tráfego.

Com o passar do tempo, o ambiente acabou crescendo e o seu gerenciamento começou a se tornar muito custoso aos funcionários do time de infraestrutura. Algumas aplicações geram uma quantidade muito alta de arquivos de *log*, requisitando limpezas repetitivas ao longo da semana. Não existe uma padronização ou calendário para a execução de atualizações de segurança no ambiente de servidores Windows, sendo assim difícil manter todo o ambiente atualizado, prevenindo falhas de segurança e possíveis ataques cibernéticos. O processo de instalação e remoção de softwares é feito de forma manual, acessando as máquinas via conexões RDP ou SSH, o que pode resultar em aumento de uso de processamento, memória e impactar aplicações mais críticas. Além disso, devido à configuração ser manual, não há uma padronização nos servidores, versionamento das modificações e documentação atualizada, resultando em retrabalho e muito tempo perdido em análise de problemas. Como a equipe é reduzida e a quantidade de demanda é muito alta, os indicadores de *Service Level Agreement* (SLA) da equipe começaram a ser

afetados, gerando avaliações negativas dos clientes e da TI.

O ambiente on-premises possui diversos servidores, bancos de dados, *appliances* e *firewalls*. Ele é interligado com a *Azure Cloud* via *Express Route*, uma conexão de alta velocidade de até 100 Gbps com a Microsoft. Já na *Azure Cloud*, a empresa possui diversos serviços, aplicações e bancos de dados em IaaS (máquinas virtuais, contêineres e kubernetes) e também em SaaS e PaaS. Podemos representar simplificadamente o ambiente com a Figura 4.1.

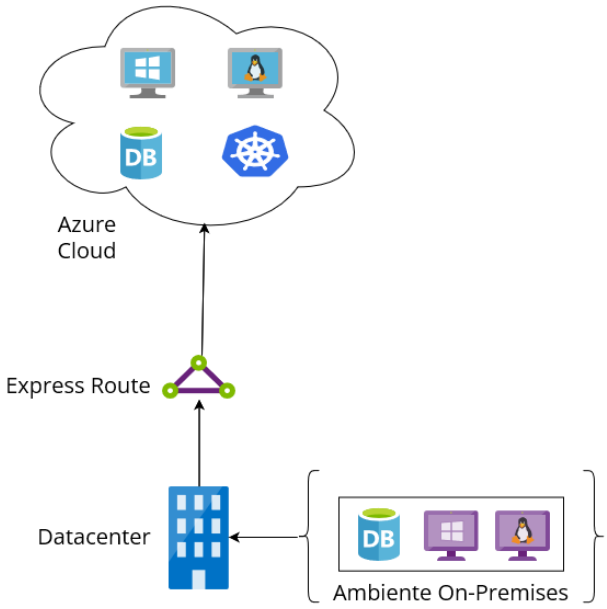

Figura 4.1: Arquitetura simplificada da infraestrutura atual

4.3 Ambiente experimental

Para resolver os problemas apresentados seção anterior, a equipe de infraestrutura decidiu fazer o uso das duas ferramentas de automação Ansible e vRx, que serão apresentadas nas próximas seções, no ambiente. As ferramentas foram implementadas no ambiente *On-premises*, pois a maior carga de trabalho da empresa está em seu *data center*, e isso reduziria a banda de rede utilizada na comunicação do ambiente *On-premises* com a nuvem. Outro peso para essa escolha foi o fato da infraestrutura local atual já possuir os recursos de computação e armazenamento, reduzindo então o custo de implementação das soluções. Podemos representar o ambiente experimental com a Figura 4.2, e abaixo vamos detalhar melhor as ferramentas que serão implementadas.

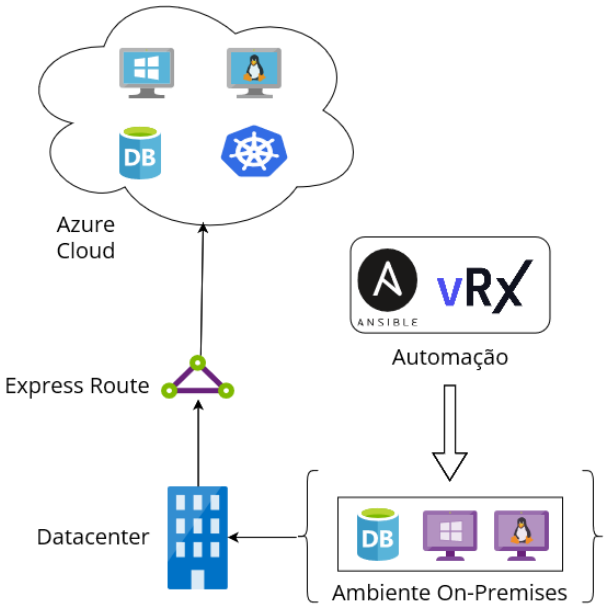

Figura 4.2: Arquitetura simplificada do ambiente experimental proposto

4.3.1 vRx

A equipe decidiu utilizar em seu ambiente de servidores Windows a ferramenta vRx, uma solução completa de avaliação e correção de vulnerabilidades e gerenciamento de atualizações de sistemas operacionais e aplicações. A ferramenta também é capaz de executar *scripts* via Powershell, possibilitando então a instalação, remoção e configuração de softwares, sem a necessidade de realizar conexões nas máquinas virtuais. O fato do software também poder ser utilizado para gerenciamento e correção de vulnerabilidades dá a equipe mais segurança em seu ambiente.

Após um trabalho inicial de instalação em todo o ambiente Windows existente On-Premises e na nuvem, e modificação do modelo Windows utilizado para novas máquinas, a aplicação vRx foi instalada em todo o ambiente, juntamente da criação de um inventário organizado e dividido por versões dos sistemas operacionais. Com a ferramenta já funcional, foi desenvolvido um calendário de atualização de sistemas operacionais em todo o parque, representado na Figura 4.3, agilizando e automatizando todo o processo de atualização e correção de *Common Vulnerabilities and Exposures* (CVEs).

Com o calendário já em funcionamento, as atualizações de segurança em máquinas Windows passaram a ser executadas através de *scripts* na aplicação vRx, conforme a Figura 4.4.

Janelas de aplicação de atualizações								
Calendário de Mês	**Grupos de servidores e estacoes de trabalho**		**Horários**					
Data	**SEM** comunicado	**COM** comunicado	7	10	12	14	18	20
2ª Segunda do mês	Grupo A			X		X	X	
3ª Segunda do mês	Grupo B			X		X	X	
2ª Quarta do mês		Grupo C	X				X	X
3ª Quarta do mês		Grupo D	X				X	X
4ª Terça do mês		Grupo exceção 1	Horário combinado					
4ª Quinta do mês		Grupo exceção 2	com a área de negócio					

Figura 4.3: Calendário de atualizações planejado pela equipe de Infra

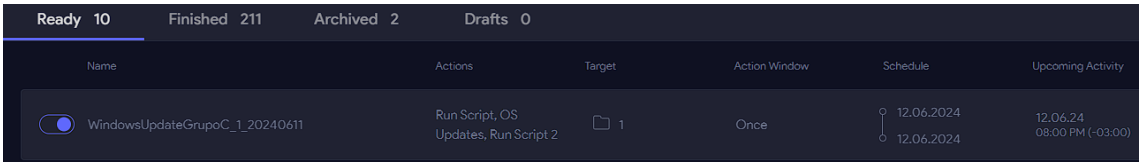

Figura 4.4: Exemplo de *script* de atualização na aplicação vRx

4.3.2 Ansible - AWX Project

Para o ambiente Linux, a equipe fez a implementação da ferramenta de automação Ansible. Devido ao fato de grande parte do parque de servidores Linux da empresa serem variações do Red Hat Enterprise Linux e também para facilitar o uso da ferramenta Ansible, a equipe decidiu fazer uso do AWX. O AWX é um projeto *open-source* patrocinado pela própria Red Hat, que permite aos seus usuários um melhor controle do Ansible, disponibilizando uma interface Web.

Foi criado um inventário de servidores Linux da empresa, organizados por sistema operacional, versão macro e micro, e ambiente. Com isso a equipe ganhou a liberdade para executar *scripts* específicos dependendo da configuração do ativo. Além disso, a equipe integrou a ferramenta ao Microsoft Entra ID (antigo Azure AD), assim possibilitando autenticação segura via *Single Sign-On* (SSO) e *Multi-factor authentication* (MFA), e com o Azure DevOps. O Azure DevOps é um conjunto de ferramentas de desenvolvimento fornecido pela Microsoft, responsável pelo suporte no desenvolvimento de softwares. Com essa integração, os códigos escritos começaram a ser versionados e armazenados no Azure Repos, o repositório de código Git e controle de versão da Microsoft.

A primeira tentativa de automatização da equipe com o Ansible foi a criação de *scripts* de limpeza de arquivos de *logs* executados em servidores produtivos ao longo da semana. O efeito foi imediato, pois o espaço em disco utilizado em vários servidores produtivos se estabilizou, reduzindo a necessidade de atuação da equipe, o número de

incidentes e a quantidade de acionamentos fora do horário comercial. A segunda iniciativa foi a criação de um *script* para alterar a senha de usuários em servidores sem a necessidade de conectar nas máquinas, tudo centralizado no Ansible, em um usuário com poderes administrativos nos servidores. A terceira iniciativa foi utilizar a *Ansible Facts*, recurso do Ansible capaz de descobrir informações de forma remotamente, e com isso atualizar a base de informações dos servidores do parque Linux. A Figura 4.5 ilustra alguns dos *scripts* criados no Ansible.

Figura 4.5: Exemplos de *scripts* desenvolvidos no Ansible

4.3.3 Zabbix e Grafana

O Zabbix é uma ferramenta de monitoramento de infraestrutura de TI de código *open-source*. A ferramenta além de monitorar ambientes é capaz de coletar dados, enviar alertas e notificações, executar *scripts*, é flexível e escalável. O Grafana é uma ferramenta de visualização de métricas, *logs* e dados de monitoramento. Com ela é possível criar painéis interativos utilizando várias fontes de dados diferentes e possui *plugins* oficiais e de terceiros para diversas aplicações de monitoramento.

A equipe sempre utilizou o Zabbix para monitorar seu ambiente, porém a versão utilizada em produção se encontra em fim de suporte. Com essa restrição, a equipe tem tido problemas na criação de monitoramentos mais atualizados, além da dificuldade de gerenciamento de seu banco de dados e lentidão no *front-end* da aplicação. A versão 6.0 da ferramenta possui uma nova versão do agente de coleta com suporte a contêineres e

kubernetes, além de novos modelos de monitoramento para a nuvem, bancos de dados e *appliances*, como visto na Figura 4.6. Devido a criticidade, um processo de migração da aplicação tem de ser feita com todo cuidado e de forma paralela até o momento de sua virada para a produção.

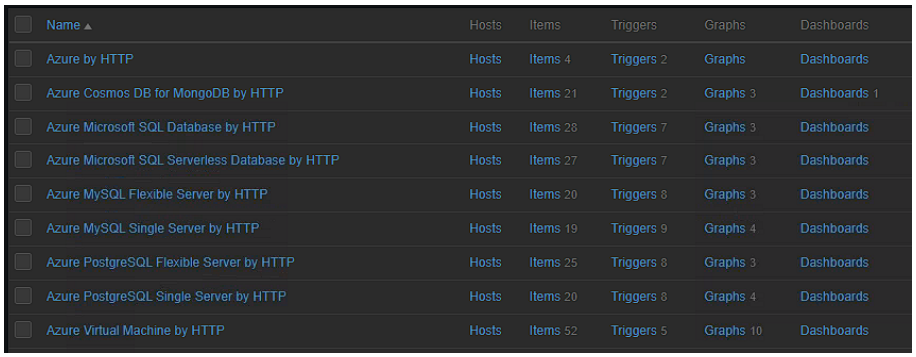

Figura 4.6: Exemplos de modelos para monitoramentos Azure no Zabbix

Foi identificado que entre as tarefas macro a serem realizadas na migração do Zabbix para uma nova versão, temos:

- Criação de um novo servidor;
- Instalação da aplicação, banco de dados e *tunning* do ambiente;
- Importação das configurações, modelos, mídias, *scripts* utilizadas no ambiente antigo;
- Replicação dos usuários, grupos de usuários e ações;
- Importação dos *hosts* e seus devidos grupos;
- Limpeza de usuários, *hosts*, monitoramentos e ações que já não estão mais sendo utilizadas;
- Instalação e reconfiguração em larga escala dos agentes Zabbix no parque de servidores.
- Replicação de regras de *firewall* presentes no ambiente antigo.

Podemos considerar a criação e instalação do Zabbix e banco de dados como uma tarefa apesar de manual, simples, pois apenas seguir a documentação oficial já o suficiente

para criar o ambiente e fazer as configurações base. O *tunning* do ambiente deve ser feito com base na quantidade de monitoramentos e frequência de coleta, podendo ser alterada se necessário. Algumas configurações do Zabbix podem ser facilmente exportadas, reduzindo bastante o tempo de configuração, porém ainda é necessário manualmente criar os devidos usuários, grupos e ações. A tarefa que de fato consumiria muito tempo na migração seria a instalação e reconfiguração dos agentes de monitoramento devido a heterogeneização do ambiente. Outro ponto importante é que apesar da migração da ferramenta, a versão em fim de vida precisa se manter operacional para coleta de relatórios de disponibilidade do ambiente e auditoria pelo período de um ano.

O Grafana também se encontrava em uma versão desatualizada e estava impactando a criação e atualização dos painéis de visualização. Sendo assim a equipe adicionou também em sua lista de pendências a sua migração, logo após o Zabbix. Uma nova versão do Grafana também seria útil em caso de utilização de novas ferramentas de monitoramento, como por exemplo o Prometheus, Solarwinds ou Datadog, além de permitir a criação de painéis de visualização personalizados para outras equipes da TI monitorarem também seus devidos servidores e sistemas com mais precisão e agilidade.

Através de uma empresa terceirizada foi realizada uma previsão de quantas horas seriam gastas para executar essa migração e o resultado foi de mais 200h de esforço, o que motivou a equipe a fazer o uso das ferramentas de automação citadas anteriormente para a implementação e maior agilidade da migração.

4.4 Implementação do Zabbix

4.4.1 Implementação no Ambiente Windows

Visando não causar impactos no ambiente como um todo, foram adotadas diferentes estratégias de instalação e configuração dos agentes do Zabbix. No ambiente Windows foi decidido que o *Zabbix Agent* antigo permaneceria em funcionamento utilizando sua porta padrão TCP 10050 e um *Zabbix Agent 2*, que possui uma nova linguagem, novos recursos e melhor desempenho, seria instalado na porta TCP 10049 apontando para o novo servidor da aplicação. Como o sistema operacional trata os dois agentes como processos

diferentes, essa estratégia evitaria que o monitoramento atual parasse de funcionar.

Para executar a instalação em massa no ambiente foi utilizado um *script* powershell executado pela ferramenta vRx, conforme a Figura 4.7. A ferramenta armazena no *script* criado o arquivo instalador e passados os devidos parâmetros de instalação, executa remotamente sua instalação. Felizmente, o Zabbix possui uma documentação rica e com os parâmetros em mão foram executadas algumas iterações do *script* em servidores não produtivos de diferentes versões de sistema operacional (2008, 2012 e 2016) para garantir que uma vez executado em massa, não causaria indisponibilidade nos servidores ou agente atual.

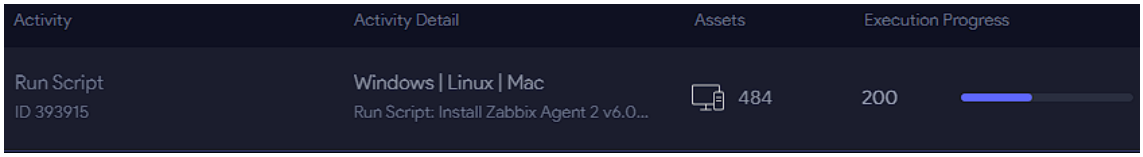

Figura 4.7: Exemplo de execução do *script* de instalação dos agentes Windows

Após os testes concluídos, o *script* foi executado em um número maior de máquinas e possuiu resultado satisfatório, ocorrendo problema em menos de 10% do parque de servidores. Apesar do valor ser alto, já era esperado devido a restrições de *firewall*, máquinas sem acesso ao domínio, entre outros problemas que seriam tratados de forma isolada no futuro.

4.4.2 Implementação no Ambiente Linux

A implementação no ambiente Linux foi mais complicada por diversos fatores. A maioria dos servidores Linux da empresa usam distribuições vindas do Red Hat Linux, o que simplifica a instalação e configuração, porém parte dos servidores são extremamente críticos e/ou se encontram em versões em *End of Life* (EOL). Com isso, não é possível acessar os repositórios oficiais para executar a instalação dos novos agentes nas máquinas.

Com isso em mente, foram adotadas três estratégias diferentes de implementação. Para as máquinas novas, foi decidido que o novo Agent seria instalado e com isso os modelos de servidores Linux foram atualizados com ambos os agentes. Para as máquinas que possuíam acesso ao repositório, foi desenvolvido um *script* de instalação do novo agente utilizando o Ansible, visto na Figura 4.8.

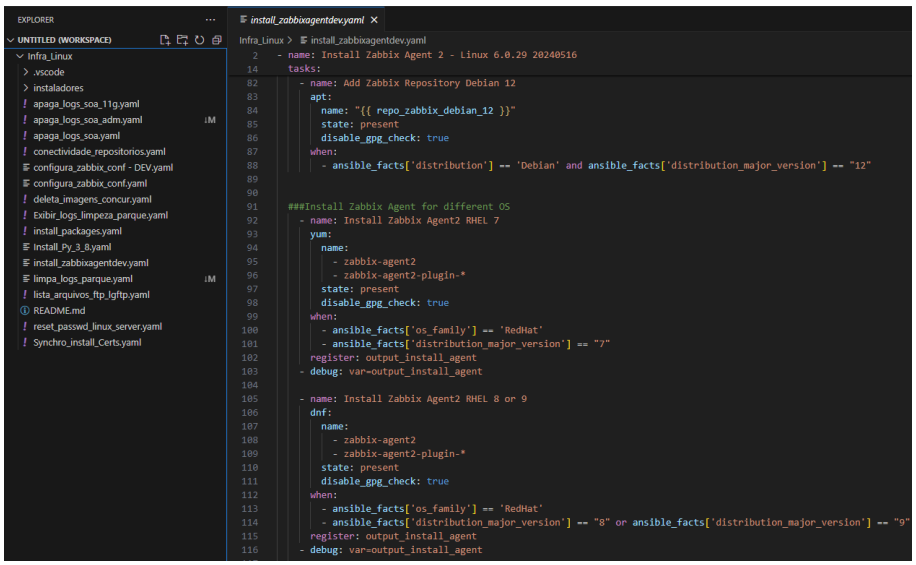

Figura 4.8: Parte do *script* de instalação do *Zabbix Agent*

Quanto as máquinas em EOL, foi adotada uma medida mais conservadora. Versões antigas do Zabbix Agent possuem compatibilidade com a nova versão da aplicação, possibilitando manter os monitoramentos mais simples. Apesar de não ser o cenário ideal, uma alteração do arquivo de configuração e reiniciar a aplicação já seriam o suficiente para o funcionamento dos monitoramentos. Essa ação causaria uma leve indisponibilidade do agente, porém dada a impossibilidade da instalação do novo agente, esse risco teria de ser tomado. Para alteração dos arquivos de configuração e foi criado um segundo *script*, mostrado na Figura 4.9, executado pelo Ansible.

Ambos *scripts* executados pelo Ansible foram previamente testados em máquinas não produtivas para identificar possíveis falhas, posteriormente foram executados em pequena escala, e foram adaptados. Após execução do *script* no parque, o resultado satisfatório e semelhante ao ambiente Windows, e poucos servidores tiveram de ser feito o processo de forma manual.

4.4.3 Desafios encontrados

As implementações descritas nas seções acima foram executadas em cerca de 6 meses e atualmente se encontram funcionais e estáveis. Entre os desafios encontrados, podemos citar:

- **Estudo das ferramentas e arquitetação**:

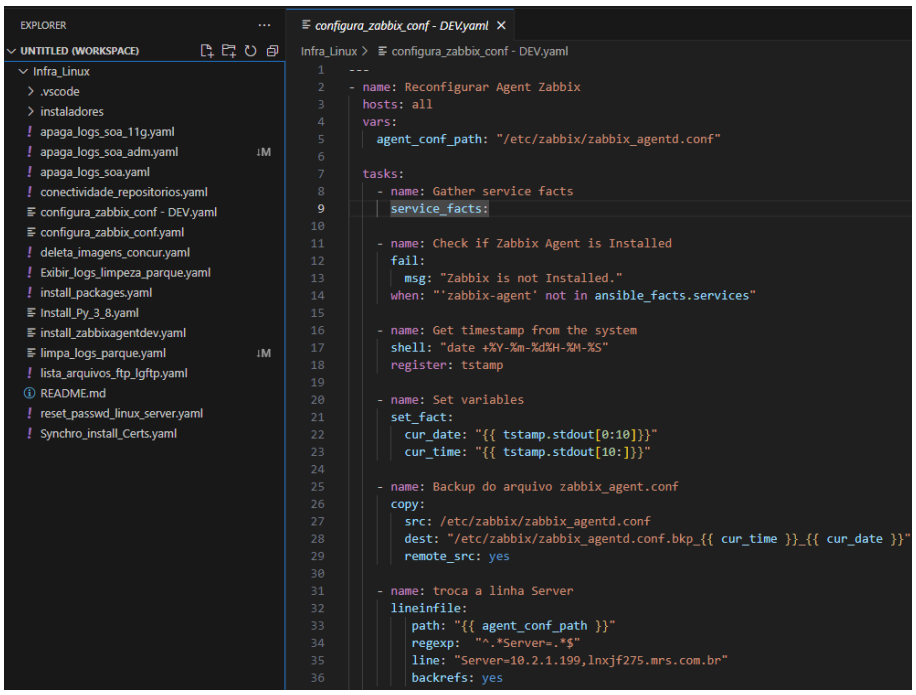

Figura 4.9: Parte do *script* de reconfiguração do *Zabbix Agent*

Foi necessário executar um estudo profundo das ferramentas utilizadas para garantir que a implementação delas seria feita corretamente no ambiente da empresa, além de entender suas funcionalidades para colocar em prática. Nesse processo foram gastos mais de 70 horas de estudo e implementação. O uso de Terraform, uma ferramenta de criação e versionamento de infraestrutura através de código com suporte *multi-cloud* e bastante utilizada no mercado, também foi cogitado como parte do projeto, porém após cerca de 50h de estudo e treinamento, foi decidido que num primeiro momento sua implementação não seria possível sem uma grande mudança de cultura na equipe.

- **Contratempos na implementação - Zabbix**:

  Foram necessários diversos *troubleshootings* relacionados a regras de *firewall* e problemas de comunicação das ferramentas com o parque de servidores devido a quantidade de ferramentas de segurança implementadas no ambiente. Além disso, durante a implementação do Zabbix, houve um problema na tabela de histórico do banco de dados da aplicação, que havia sido migrado a partir da versão atual em produção. Como a nova versão do Zabbix trabalha de forma diferente, os novos dados começaram a corromper a tabela, e a melhor forma de resolver foi recriando

como um todo o banco e importando somente os dados do *front-end* produtivo, o que acabou gerando um grande retrabalho.

- **Higienização do novo ambiente - Zabbix**:

  O Zabbix antigo possuía diversos monitoramentos que já não eram mais utilizados, além de servidores que já haviam sido desativados permanecerem na base de dados (desabilitados). Foi necessário um extenso trabalho de verificação e identificação dos monitoramentos e suas ações, além da higienização da base de dados e servidores previamente desativados. Esse processo teve de ser feito com muita cautela, pois é indispensável que todos os monitoramentos se mantivessem.

- **Falta de documentação - Zabbix**:

  Ao analisar os monitoramentos e servidores, foram descobertas algumas alterações feitas na aplicação que não foram devidamente documentadas no passado. Foi necessário um grande esforço para identificar essas alterações e replicar no novo ambiente, que possui um sistema operacional diferente do antigo e muito mais atualizado.

- **Problemas de compatibilidade**:

  A nova versão do Zabbix modificou a forma com que a base de dados é conectada com o Grafana, sendo assim tivemos contratempos na configuração da fonte de dados. Além disso, alguns monitoramentos tiveram seus parâmetros alterados na nova versão e tiveram de ser alterados. Quanto ao Ansible, é um pré-requisito que as máquinas Linux tenham o Python 2 ou 3 instalados. Como algumas máquinas estavam em EOL, não foi possível sua instalação e inclusão na automação.

- **Problemas nos agentes e instalação - Zabbix**:

  Foram encontradas algumas incompatibilidades do agente com versões de antivírus legado em alguns servidores, necessitando diversos testes para identificar a causa raiz. Em alguns casos, o agente não conseguiu ser reinicializado, sendo necessário negociações com as áreas clientes para reinicialização do sistema operacional. Em alguns casos, o sistema operacional não dava suporte a versão do agente e tivemos

de adotar medidas paliativas, como a utilização do agente em uma versão um pouco mais antiga.

- **Instalações em ambientes críticos - Zabbix**:

  Parte dos servidores eram críticos para a operação da empresa, com necessidade de janela de manutenção e parada da operação para chaveamento. Foi necessário aguardar meses para a disponibilização de uma janela de instalação dos agentes, atrasando a implementação do Zabbix em parte crítica do parque.

4.5 Resultados

O processo de implementação das ferramentas foi considerado um sucesso pela equipe e seu uso tem produzido ótimos resultados. O vRx executa atualizações mensais nos sistemas operacionais Windows, resolvendo diversos CVEs e mantendo o ambiente devidamente atualizado e mais seguro. Ele também tem sido utilizado para instalação e remoção de softwares em larga escala no ambiente Windows. Conforme a Figura 4.10 abaixo, o Ansible executa de forma automatizada diversas tarefas ao longo da semana e isso tem reduzido a quantidade de incidentes e acionamentos fora do período comercial. Com isso, a equipe agora gasta menos tempo com a sustentação do ambiente e passa mais tempo desenvolvendo novos *scripts* e cuidando de novos projetos internos e para a TI. O resultado foi tão satisfatório que o coordenador da equipe designou um dos funcionários a ficar em tempo integral focado em trazer mais melhorias ao Ansible e criação de novas automações.

O Zabbix se encontra em processo final de migração, aguardando somente a resolução de comunicação de alguns agentes e chaveamento do ambiente crítico para instalação em alguns servidores. Conforme vemos na Figura 4.11 abaixo, o Zabbix já possui dezenas de milhares de itens sendo coletados, com cerca de 900 novos valores sendo coletados por segundo. Apesar de ainda não ser considerado produtivo, ele já é utilizado para verificar o desempenho dos servidores, além de já possuir novos monitoramentos personalizados. A previsão é de que a migração seja finalizada no segundo semestre do ano de 2024.

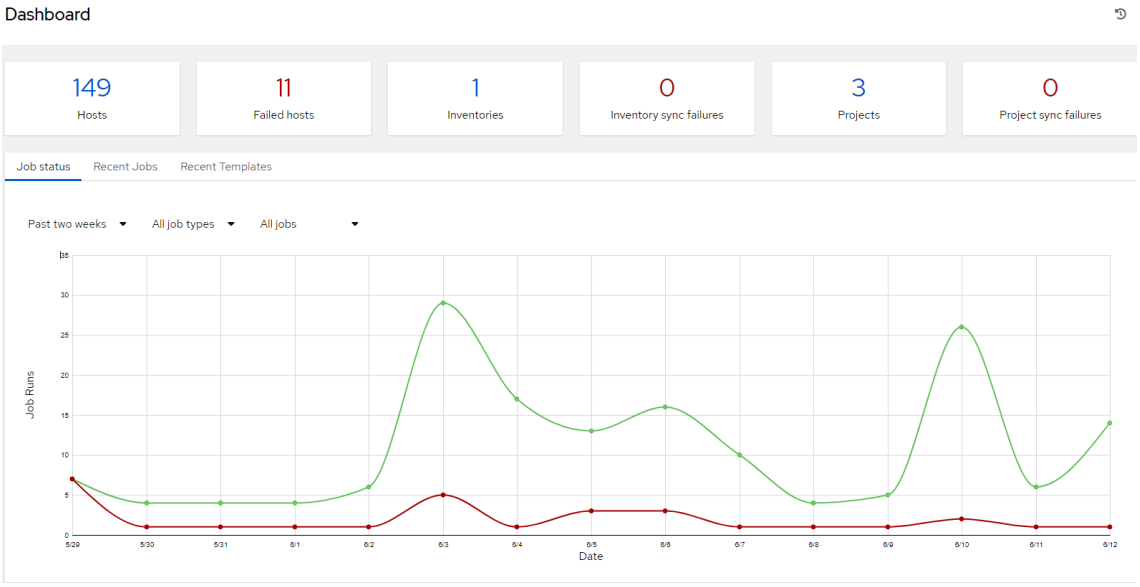

Figura 4.10: Execuções de *scripts* Ansible no ambiente

System information

Parameter	Value	Details
Number of hosts (enabled/disabled)	681	659 / 22
Number of templates	117	
Number of items (enabled/disabled/not supported)	82527	65518 / 11475 / 5534
Number of triggers (enabled/disabled [problem/ok])	21781	15323 / 6458 [125 / 15198]
Number of users (online)	79	1
Required server performance, new values per second	911.51	
High availability cluster	Disabled	

Figura 4.11: Informações sobre o sistema Zabbix

4.6 Considerações do Capítulo

Nesse capítulo foi apresentada a proposta de utilização de ferramentas de automação para gerenciamento de infraestrutura e implementação de uma ferramenta de monitoramento em grande escala. As ferramentas de automação vRx e Ansible foram apresentadas, e além disso, mostramos que com elas em ação, foi possível executar diversas tarefas automatizadas no ambiente, auxiliando a equipe de infraestrutura no seu dia a dia. Foi apresentado o processo de implementação do Zabbix com uso de automação e por fim, os desafios encontrados durante toda a implementação das ferramentas e Zabbix foi relatado.

5 Conclusões e trabalhos futuros

5.1 Conclusões

O trabalho foi realizado em um ambiente corporativo onde boa parte da execução de tarefas de gerenciamento da infraestrutura eram feitas de forma manual, consumindo bastante esforço da equipe de infraestrutura de TI no dia a dia. Com a adoção do ambiente experimental, foi possível trazer automação a boa parte das atividades, auxiliando no enfrentamento dos desafios relatados pela equipe, além de implementar em larga escala uma aplicação de monitoramento no ambiente.

Concluída a implementação do ambiente experimental, podemos dizer que o processo de implementação e uso de infraestrutura como código para gerenciamento de uma infraestrutura de larga escala não é simples. É necessário muito estudo e cuidado na sua arquitetação e implementação, garantindo que o código seja limpo e documentado, facilmente escalável e adaptável. É importante manter uma gestão cuidadosa e descentralizada, com devido versionamento e seguindo as melhores práticas de segurança. É necessário também uma mudança geral de cultura da equipe, pois a forma de se administrar o ambiente tende a ser menos manual, porém com maior complexidade de implementação. É fundamental que essa mudança de cultura aconteça gradativamente, com devida documentação e treinamento, porém o seu resultado é extremamente efetivo. Uma vez solidificado, os códigos não tem a necessidade de ser alterados com frequência e o tempo que antes era gasto com tarefas repetitivas pode ser utilizado em novas soluções inteligentes e demais melhorias.

5.2 Trabalhos Futuros

Devido a amplitude do tema, a proposta do trabalho teve de ser reduzida para garantir que sua execução fosse possível no intervalo de tempo disponível. Visto que o resultado da proposta foi positivo, existem melhorias que podem ser feitas nas ferramentas aqui

utilizadas. O primeiro passo seria a expansão do uso do Ansible também para os servidores de sistema operacional Windows. Com isso, seria possível centralizar em apenas uma ferramenta toda a configuração de ativos de infraestrutura, mantendo o vRx somente para aplicação de atualizações de segurança. O Ansible também pode ser utilizado em conjunto com o Azure Devops para criação de pipelines de execução automatizados na nuvem e existe nesse momento um estudo sobre sua viabilidade.

Durante o trabalho foi cogitada implementação do Terraform e foram gastas cerca de 50 horas de estudos e treinamento, porém sua implementação se tornou inviável em tão pouco tempo devido a alta demanda da equipe e necessidade de modificação da cultura interna. Uma outra ideal é utilizar o Terraform integrado com o Azure Pipelines e Repos para centralizar a criação e alteração de infraestrutura na Azure. Recentemente adquirido pela IBM, o Terraform deixou de ser uma ferramenta *open-source*, motivando a criação de um fork *open-source* chamado OpenTofu. É interessante então, talvez utilizar o OpenTofu, que até então é bem semelhante do Terraform e irá se manter sem custos. É possível também, visto que a empresa possui ambiente na Azure, utilizar para criação de infraestrutura o Bicep, linguagem específica de domínio da Microsoft utilizada no *back-end* da Azure.

A empresa atualmente utiliza o ServiceNow como seu ITSM e CMDB, sendo assim é possível realizar sua integração com o Zabbix, para automatizar a abertura de chamados com as devidas informações do ativo diretamente para as equipes responsáveis. É possível também, fazer sua integração com o Azure DevOps para executar pipelines de acordo com a solicitação requisitada sem a necessidade ação humana.

Bibliografia

BASS, L.; WEBER, I.; ZHU, L. *DevOps: A Software Architect's Perspective*. 1st. ed. [S.l.]: Addison-Wesley Professional, 2015. ISBN 0134049845.

DONNO, M. D.; TANGE, K.; DRAGONI, N. Foundations and evolution of modern computing paradigms: Cloud, iot, edge, and fog. *IEEE Access*, v. 7, p. 150936–150948, 2019.

FISHER, C. Cloud versus on-premise computing. *American Journal of Industrial and Business Management*, v. 8, n. 9, 2018. Disponível em: ⟨https://www.scirp.org/journal/cta?paperid=87661⟩.

GUTTA, L. M. Achieving operational excellence in cloud management: Practical evaluation of infrastructure as code and the well-architected framework's adoption to improve process maturity. *International Journal of Managment Education for Sustainable Development*, v. 6, n. 6, p. 1–19, 2023. ISSN 3246-547X. Disponível em: ⟨https://www.ijsdcs.com/index.php/IJMESD/article/view/464⟩.

HASAN, M. R.; ANSARY, M. S. Cloud infrastructure automation through iac (infrastructure as code). *International Journal of Computer (IJC)*, v. 46, n. 1, p. 34–40, Feb. 2023. Disponível em: ⟨https://ijcjournal.org/index.php/InternationalJournalOfComputer/article/view/2043⟩.

KUMARA, I. et al. The do's and don'ts of infrastructure code: A systematic gray literature review. *Information and Software Technology*, v. 137, p. 106593, 2021. ISSN 0950-5849. Disponível em: ⟨https://www.sciencedirect.com/science/article/pii/S0950584921000720⟩.

MELL P.; GRANCE, T. The nist definition of cloud computing. *National Institute of Standards & Technology*, Gaithersburg, MA, USA, p. 2, 2011. Disponível em: ⟨https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-145.pdf⟩.

MORRIS, K. *Infrastructure as Code: Dynamic Systems for the Cloud Age*. O'Reilly, 2021. ISBN 9781098114671. Disponível em: ⟨https://books.google.com.br/books?id=Wz2KzQEACAAJ⟩.

QUATTROCCHI, G.; TAMBURRI, D. A. Infrastructure as code. *IEEE Software*, v. 40, n. 1, p. 37–40, 2023.

SOUZA, I. S. e.; FRANCO, D. P.; SILVA, J. P. S. G. Infrastructure as code as a foundational technique for increasing the devops maturity level: Two case studies. *IEEE Software*, v. 40, n. 1, p. 63–68, 2023.

VERGADIA, P. *Visualizing Google cloud: 101 Illustrated references for cloud engineers and architects*. [S.l.]: John Wiley & Sons, 2022.

ZHANG, S. et al. Practical adoption of cloud computing in power systems—drivers, challenges, guidance, and real-world use cases. *IEEE Transactions on Smart Grid*, v. 13, n. 3, p. 2390–2411, 2022.