SOLUÇÃO INTEGRADA PARA CONFIGURAÇÃO AUTOMATIZADA DE ATIVOS DE REDE

AÉCIO S. PIRES, PAULO DITARSO MACIEL JUNIOR, DIEGO ERNESTO ROSA PESSOA (IFPB, Campus João Pessoa)

E-mails: aeciopires@gmail.com,paulo.maciel@ifpb.edu.br,diego.pessoa@ifpb.edu.br.
Área de conhecimento:(Tabela CNPq): 1.03.03.04-9 Sistemas de Informação.
Palavras-Chave: ativos de rede; automação; versionamento; infraestrutura como código.

1 Introdução

As operações de TI estão cada vez mais imbuídas de técnicas ágeis e sucintas que buscam encurtar o ciclo de desenvolvimento de um software ou unificar seus procedimentos com as atividades de desenvolvimento. Essa tendência de usar recursos da engenharia de software para reduzir o espaço, tempo e esforço entre atividades de desenvolvimento e operações de software, bem como a distância técnica e organizacional entre os dois tipos de equipes responsáveis é conhecida como cultura *DevOps* (ARTA et al., 2017). Como parte de seus princípios, muitas práticas envolvem a reutilização de padrões e sistemas de software, tais como o versionamento e a revisão de código.

Dentro do contexto da cultura DevOps, a gerência de **Infraestrutura como Código** (IaC, do inglês *Infrastructure as Code*) é uma abordagem que permite a automação de uma infraestrutura utilizando práticas de desenvolvimento de software, enfatizando rotinas consistentes e repetitivas para orquestração, provisionamento e configuração (MORRIS, 2016). Além da automação, o fato de o ambiente ter sido especificado em formato de código significa explicitamente que o mesmo poderá ser implantado em qualquer lugar, minimizando a possibilidade de inconsistências (JIANG; ADAMS, 2015).

Esta pesquisa propõe a integração de sistemas de software existentes para a gerência de ativos de redes, utilizando uma abordagem de Infraestrutura como Código. A solução proposta, denominada PipeConf, centraliza a automação das atividades de configuração e permite realizar o backup de todo o processo. Além disso, a mesma busca abstrair a diversidade de sintaxes para diferentes ativos, bem como possibilita o gerenciamento de uma grande quantidade de dispositivos. Em resumo, a solução proposta pode ser vista como um *pipeline* escrito como código, que consiste em um conjunto de sistemas de software, fluxos e processos automatizados para integrar um conjunto de componentes de software (INDIA, 2019).

2 Materiais e Métodos

A arquitetura de *software* integrado proposta, denominada **PipeConf** (Figura 1), provê a automatização da configuração de ativos de rede heterogêneos utilizando a abordagem de IaC. A arquitetura compreende um ***Plano de Dados*** e um ***Plano de Controle***. O primeiro tem a responsabilidade de armazenar e versionar os arquivos contendo as credenciais e configurações dos ativos de rede; enquanto o último tem a responsabilidade de executar funções e processos necessários ao gerenciamento dos mesmos.

No Plano de Dados está o **Módulo de Versionamento** (MV), que coordena o registro das versões do código produzido e o armazenamento das configurações dos ativos (*backup*). Agregados no Plano de Controle estão: o **Módulo de Criptografia** (MC), responsável pela (de)criptografia dos arquivos de credenciais e configurações dos ativos; o **Módulo de Gerência de Configuração** (MGC), que interage com os ativos para fazer o *backup* e/ou aplicar alterações nas configurações; o **Módulo de Notificações** (MN), que envia mensagens aos administradores; e o **Módulo de Integração Contínua** (MIC), que executa o *pipeline* integrando todas as ferramentas utilizadas.

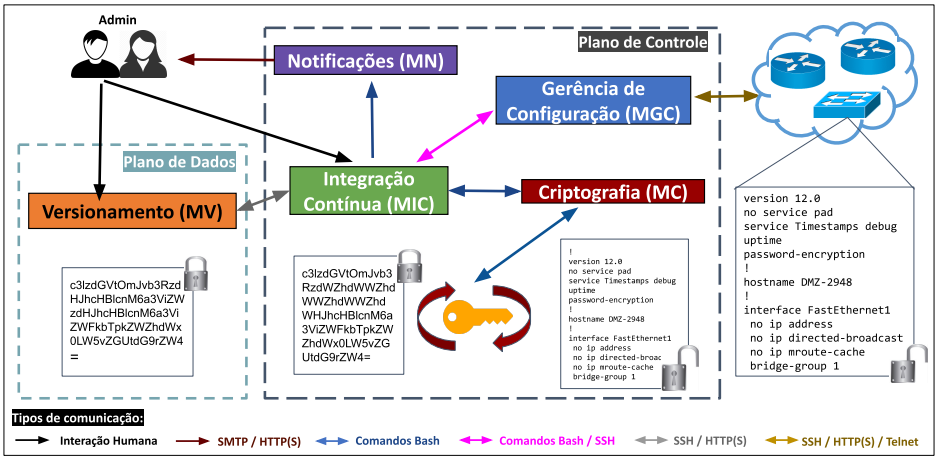

Figura 1: Arquitetura do PipeConf.

O PipeConf realiza o conjunto de 12 atividades apresentadas na Tabela 1. Não foi encontrada uma solução existente que realize todas ou a maioria dessas atividades. Além disso, fatores como a ausência de licenças gratuitas, o curto tempo de avaliação e a falta de usabilidade se mostraram impeditivos para o uso de grande parte das 10 ferramentas investigadas[1]. Dentre elas, o uso do **Unimus**[2] se mostrou viável, apesar de executar apenas as atividades **AT05** e **AT07**. O Unimus é um software utilizado para gerenciar backups e configurações de ativos de rede. A comparação entre o PipeConf e o Unimus é importante para apresentar evidências da eficiência do PipeConf. Apesar da atividade **AT07** ser considerada genérica, ressalta-se que qualquer configuração é possível ser executada nos ativos.

Atividades	Descrição
AT01	Obter do repositório Git os arquivos de credenciais e configuração.
AT02	Realizar descriptografia dos arquivos de credenciais e configuração.
AT03	Efetivar a detecção dos ativos conectados.
AT04	Obter informações sobre o modelo de cada ativo conectado.
AT05	Fazer *backup* da memória (*running-config*) antes da alteração.
AT06	Fazer *backup* da inicialização (*startup-config*) antes da alteração.
AT07	Alterar configuração do servidor NTP (a.ntp.br).
AT08	Fazer *backup* da memória (*running-config*) depois da alteração.
AT09	Fazer *backup* da inicialização (*startup-config*) depois da alteração.
AT10	Aplicar criptografia nos arquivos de *backup*.
AT11	Versionar os arquivos de *backup* no Git.
AT12	Enviar notificações da execução do PipeConf aos administradores.

Tabela 1: Descrição das atividades executadas pelo PipeConf.

3 Resultados

Durante a avaliação do PipeConf, foram realizadas as seguintes análises: **Análise de Perfilamento**, **Análise de Desempenho**, **Análise de Escalabilidade** e a **Análise de Vulnerabilidades de Segurança**, detalhadas a seguir. O ambiente computacional para execução dos experimentos, os sistemas de software utilizados em cada módulo do PipeConf e os procedimentos de instalação do ambiente de experimentação estão disponíveis na documentação do PipeConf[3].

Na **Análise de Perfilamento** foram perfilados os tempos de execução de cada atividade executada pelo PipeConf e o Unimus com até 32 ativos. Um importante destaque foi o tempo médio total relativamente baixo (menos de 10 minutos para gerenciar 32 ativos). A comparação possibilitou detectar que o Unimus apresenta

[1] https://bit.ly/3cCfqWB
[2] https://unimus.net
[3] https://gitlab.com/aeciopires/pipeconf

tempos de execução das atividades bem mais estáveis que o PipeConf. O motivo disso é que o Unimus inicializa um processo paralelo para cada ativo e deixa o sistema operacional responsável por gerenciar tais processos. Já o PipeConf paraleliza os processos de acordo com a quantidade de núcleos de CPU disponíveis. Esse ponto de melhoria será abordado em trabalhos futuros.

Na **Análise de Desempenho**, repetiu-se os mesmos experimentos com até 128 ativos. Como esperado, o tempo médio total para o PipeConf realizar as atividades aumentou. Contudo, este aumento não seguiu a mesma proporção do acréscimo na quantidade de ativos, indicando um ganho proporcional à medida que mais ativos eram gerenciados. O tempo médio total para executar as atividades com 128 ativos foi de 58,13 minutos. Esse tempo não é 128 vezes maior que o necessário para gerenciar um único ativo (2,61 minutos). O ganho proporcional foi de aproximadamente 83%. Este comportamento é intuitivo e se deve ao fato do PipeConf conseguir executar algumas atividades em paralelo, embora de maneira limitada.

Na **Análise de Escalabilidade** foi monitorado o consumo dos recursos computacionais pelo PipeConf durante todos os experimentos, totalizando 77h. Neste período, o consumo de CPU, memória e a quantidade de contêineres em execução ficaram estáveis.

Também foi realizada uma avaliação qualitativa, a **Análise de Vulnerabilidades de Segurança** do Pipeconf (no seu código e no código produzido). Nela buscou-se verificar a existência de 7 vulnerabilidades de segurança no código definidas por Rahman, Parnin e Williams (2019). Essas vulnerabilidades são padrões de codificação que podem indicar falhas de segurança conhecidas que podem ser exploradas por um atacante possibilitando a execução de atividades maliciosas e/ou obtenção privilégios. Para este fim, foram utilizadas ferramentas consolidadas no mercado [4] [5] [6] [7]. A análise não identificou a presença das vulnerabilidades, o que significa que o uso do PipeConf não adiciona vulnerabilidades de segurança conhecidas na rede em que for instalado.

4 Considerações Finais

Os resultados obtidos demonstraram um desempenho satisfatório do PipeConf, mesmo diante de um grande número de ativos gerenciados. Também foi possível verificar por meio de uma análise qualitativa que o PipeConf não adiciona vulnerabilidades de segurança conhecidas na rede em que for instalado. Como trabalhos futuros, será investigado o uso de outro componente de software para paralelizar o gerenciamento de ativos, além de uma alternativa para criptografar arquivos de maneira mais rápida. Uma análise mais extensiva da solução envolvendo experimentos empíricos com ativos de modelos e fabricantes distintos também pode ser realizada.

Referências

ARTA, M. et al. DevOps: Introducing Infrastructure-as-Code. In: *Proceedings of the 39th International Conference on Software Engineering Companion*. IEEE Press, 2017. (ICSE-C '17), p. 497–498. ISBN 9781538615898. Disponível em: <https://doi.org/10.1109/ICSE-C.2017.162>.

INDIA, S. What is devops pipeline? simplified in 200 words. In: . [s.n.], 2019. Disponível em: <https://medium.com/tech-in-200-words/what-is-devops-pipeline-simplified-in-200-words-cb524f94842c>.

JIANG, Y.; ADAMS, B. Co-Evolution of Infrastructure and Source Code: An Empirical Study. In: *Proceedings of the 12th Working Conference on Mining Software Repositories*. IEEE Press, 2015. (MSR '15), p. 45–55. ISBN 9780769555942. Disponível em: <https://doi.org/10.1109/MSR.2015.12>.

MORRIS, K. *Infrastructure as Code - Managing Servers in the Cloud*. 1st. ed. Sebastopol, CA, USA: O'Reilly Media, Inc., 2016. ISBN 978-1491924358.

RAHMAN, A.; PARNIN, C.; WILLIAMS, L. The seven sins: Security smells in infrastructure as code scripts. In: *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 2019. p. 164–175. ISBN 9781728108698. Disponível em: <https://ieeexplore.ieee.org/document/8812041>.

[4] Codeac: https://www.codeac.io
[5] Codacy: https://codacy.com
[6] Sonarqube: https://www.sonarqube.org
[7] ShellCheck: https://github.com/koalaman/shellcheck