PipeConf: Uma Arquitetura Integrada para Configuração Automatizada de Ativos de Rede Heterogêneos

Aécio S. Pires[1], Paulo D. Maciel Jr.[1], Diego Pessoa[1], Fernando Matos[2], Aldri Santos[3]

[1]Unidade Acadêmica de Informática - Instituto Federal da Paraíba (IFPB)
[2]Centro de Informática – Universidade Federal da Paraíba (UFPB)
[3]Depto. de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG)

aeciopires@gmail.com, {paulo.maciel,diego.pessoa}@ifpb.edu.br
fernando@ci.ufpb.br, aldri@dcc.ufmg.br

Abstract. *The manual management of network assets is susceptible to configuration errors, lack of standardization, a large amount of repetitive work and little or no traceability of changes over time. The Infrastructure as Code (IaC) approach makes it possible to automate the process of configuring resources such as operating systems, network services, containers and applications, by treating them as software and allowing standardization and configuration rollback. This paper proposes an integrated architecture based on different software tools that use the IaC approach, in order to automate the configuration of network assets, considering different models and manufacturers. A quantitative analysis shows the architecture's efficiency in response time and scalability. The architecture achieved a proportional gain of 83% in the average response time to manage 128 assets, with no significant increase in processing and memory usage.*

Resumo. *O gerenciamento manual de ativos de rede está suscetível a erros de configuração, falta de padronização, grande quantidade de trabalho repetitivo e pouca ou nenhuma rastreabilidade de alterações ao longo do tempo. A abordagem Infrastructure as Code (IaC) permite automatizar o processo de configuração de recursos como sistemas operacionais, serviços de rede, contêineres e aplicações, tratando-os como software e possibilitando a padronização e reversão da configuração. Este artigo propõe uma arquitetura integrada a partir de ferramentas de softwares diferentes e que usam a abordagem IaC, a fim de automatizar a configuração de ativos de rede, considerando diferentes modelos e fabricantes. Uma avaliação quantitativa mostra a eficiência da arquitetura em tempo de resposta e escalabilidade. A arquitetura obteve um ganho proporcional de 83% no tempo médio de resposta para gerenciar 128 ativos, sem aumento significativo no uso de processamento e memória.*

1. Introdução

Um provisionamento bem planejado e uma gestão eficiente dos recursos de TI de uma empresa contribuem para alcançar vantagens competitivas em termos de negócios [Christopher 2019]. Uma parte fundamental da gerência envolve a configuração de ativos de rede, tais como: *switches*, roteadores, *firewalls* e pontos de acesso. A medida que aspectos importantes como conectividade, desempenho, confiabilidade e segurança de uma rede dependem dessa gerência, o comportamento apropriado dos ativos definido por suas respectivas configurações faz-se imprescindível no dia a dia das corporações [Liu et al. 2018]. Entretanto, gerenciar o ciclo de vida das configurações destes ativos

é notoriamente desafiador aos administradores de rede, pois envolve atividades que vão desde a geração e atualização de arquivos de configurações, passando pelo diagnóstico de problemas e transição de estados. Além disso, essas tarefas muitas vezes são realizadas de maneira manual, por meio da execução de comandos diretamente nos ativos.

Um estudo encomendado pela Cisco em 2016 [Shah and Dubaria 2019] apontou que cerca de 95% das alterações na configuração de ativos de rede eram feitas manualmente, 70% das violações de políticas foram ocasionadas por erros humanos e 75% dos custos de operação eram gastos com alterações e *throubleshooting*. Cerca de US$ 60 bilhões foram gastos com mão de obra e aquisição de ferramentas de operações de rede naquele ano. Outra questão relevante foi o tempo médio de 1 a 5 horas para resolver um problema após sua notificação. Por fim, ainda segundo o estudo, 74% das operadoras relataram alterações na rede que impactaram significativamente em seus negócios, das quais 97% admitiram que isso ocorreu a partir de erros humanos que causaram interrupções na rede e 22% das interrupções não planejadas foram causadas por tais erros.

A *"softwarização"* de redes, em particular a abordagem de Infraestrutura como Código (IaC, do inglês *Infrastructure as Code*), permite a automação de dispositivos utilizando práticas de desenvolvimento de *software*, enfatizando rotinas consistentes e repetitivas para provisionamento, configuração e orquestração [Morris 2016]. Além da automação, o ambiente ser especificado em formato de código significa que o mesmo poderá ser implantado em qualquer lugar, minimizando a possibilidade de inconsistências [Jiang and Adams 2015]. Contudo, usar ferramentas de IaC não representa muitas vezes uma transição simples e direta. Cada ferramenta possui uma Linguagem de Domínio Específico (DSL, do inglês *Domain Specific Language*) própria e, considerando infraestruturas com ativos heterogêneos em que são necessárias várias delas, o aprendizado de mais de uma linguagem é inevitável. Logo, um ambiente com diferentes fornecedores, modelos e/ou sistemas operacionais, implica em várias sintaxes de configuração distintas [Ismail et al. 2018]. Além disso, são poucas as ferramentas que conseguem gerenciar ativos heterogêneos. Ao considerar tais dificuldades, percebe-se a demanda por uma solução eficiente para gerenciar configurações automatizadas de ativos de rede heterogêneos.

Este trabalho propõe a integração de ferramentas existentes de IaC através da arquitetura denominada **PipeConf**. Com ela é possível centralizar a automação das atividades de *backup* e configuração dos ativos, além de utilizar políticas que permitem mudanças no comportamento do sistema sem recompilar ou reiniciar o mesmo. A arquitetura ainda abstrai a diversidade de sintaxes para diferentes modelos, facilitando a interoperabilidade e o gerenciamento de uma grande quantidade de dispositivos. Uma prova de conceito da arquitetura foi desenvolvida e uma avaliação quantitativa analisou a eficiência no gerenciamento da configuração de até 128 ativos de rede. Os resultados mostraram tempos médios de configuração satisfatórios, como por exemplo 58 minutos para configurar 128 dispositivos, o que representa um ganho proporcional de 83% em relação ao tempo de um único ativo. Adicionalmente, a solução apresentou um consumo estável dos recursos de CPU e memória, mesmo com aumento no número de ativos gerenciados.

O restante do artigo está organizado da seguinte maneira: a Seção 2 apresenta os trabalhos relacionados; a Seção 3 descreve a arquitetura do PipeConf, as políticas e os fluxos de atividades; a Seção 4 detalha a metodologia de avaliação aplicada e os resultados alcançados; e a Seção 5 apresenta as conclusões do trabalho.

2. Trabalhos Relacionados

Em [Jiang and Adams 2015], os autores apresentam um estudo empírico sobre o uso da abordagem IaC em 265 repositórios do projeto OpenStack. Eles observaram que a introdução de erros de configuração é potencializada pelo fato dos arquivos serem grandes e receberem alterações frequentes. Os arquivos de testes automatizados são fortemente acoplados entre os projetos e devem ser alterados a cada mudança nas especificações da infraestrutura. O trabalho, contudo, não trata especificamente da gerência de configuração de *switches* e roteadores. Outra diferença importante é que a arquitetura proposta neste artigo possui baixo acoplamento entre os componentes de *software* integrados.

Em [Duplyakin et al. 2015], os autores relataram a experiência no gerenciamento de um *cluster* Linux com 20 nós utilizando IaC. O trabalho destaca que os recursos de segurança e a capacidade de gerenciamento do protótipo são adequados para ambientes de produção. Entretanto, o artigo também não aborda a gerência de configuração de *switches* e roteadores. Em [Vilalta et al. 2020], por outro lado, uma experiência prática no uso de ferramentas de programação para controlar e monitorar ativos de rede é apresentada. Foram exploradas linguagens de modelagem de dados e protocolos de controle/monitoramento de redes. Contudo, presume-se que a utilização das soluções sugeridas em um processo não holístico de configuração dos ativos pode resultar em uma curva de aprendizado com tempo não desprezível. Além disso, diferente do proposto neste trabalho, um sistema de controle de versão para rastrear as mudanças no código não é abordado.

Em [Chen et al. 2018], os autores relatam que a proliferação de DSLs em redes de computadores permite explorar a capacidade de programação das redes, mas não têm sido amplamente adotadas pois apresentam uma curva de aprendizado maior entre operadores que não são programadores treinados. Além disso, novas aplicações SDN (do inglês *Software-Defined Networking*) geralmente "confiam" na funcionalidade de redes herdadas que não podem ser facilmente migradas ou analisadas. Para enfrentar tais desafios, os autores propuseram a ferramenta Facon que gera automaticamente programas em DSLs arbitrárias, com base em exemplos de entrada/saída. Como tais exemplos se aplicam a qualquer protocolo de rede, essa abordagem pode ser generalizada, permitindo migrar redes herdadas para novas DSLs ou transformar uma DSL em outra. Contudo, o trabalho não aborda o uso de ferramentas de gerência de configuração, infraestrutura como código, integração contínua e versionamento de código dos exemplos gerados pelo Facon.

Em [Opara-Martins et al. 2016], os autores descrevem a dependência de fornecedores como uma grande barreira à adoção da computação em nuvem, devido à falta de padronização. Os esforços atuais para resolver tal dependência são predominantemente orientados à tecnologia. Os autores identificaram os principais fatores de risco que causam dependência de fornecedores e os resultados demonstram a importância da interoperabilidade, portabilidade e padrões na computação em nuvem. Para mitigar os riscos, foram propostas estratégias relacionadas a contratos e seleção de fornecedores que suportem um padrão comum de estrutura de dados, protocolos e APIs. Apesar de não tratar especificamente da dependência de fornecedores de *switches* e roteadores, essa questão também faz parte do cotidiano da gestão de TI. Portanto, a arquitetura proposta neste artigo busca reduzir os efeitos desta dependência a partir da abstração de diferentes sintaxes de comandos na configuração de ativos de rede heterogêneos. De maneira similar ao que os autores propõem em [Netto et al. 2017], onde é descrita uma arquitetura de replicação de

estado de contêineres para efetuar uma coordenação como serviço, com um acoplamento leve e simples à aplicação. Contudo, enfatiza-se novamente que nenhum dos trabalhos avaliados tratam sobre a gerência de configuração de ativos de rede.

Em relação ao uso de políticas de gerenciamento, observou-se também a ausência de trabalhos no contexto de ativos de rede, embora tenham sido aplicadas em outros cenários. Como por exemplo em [Sette et al. 2019], onde políticas de alto nível controlam o acesso único a múltiplos provedores de nuvens heterogêneas; ou em [Cox et al. 2017, Oliveira et al. 2018], onde são propostas políticas de controle e priorização de tráfego em redes SDN. No entanto, ao que tudo indica, nenhum outro trabalho aborda a gerência da configuração de ativos heterogêneos a partir de uma arquitetura integrada de IaC, com versionamento de código e políticas que automatizam o processo.

De forma geral, os trabalhos descritos apresentam cinco características funcionais encontradas na arquitetura proposta nesta pesquisa: aborda conceitos e/ou ferramentas de IaC; propõe solução para gerência de configuração; aplica versionamento de *software*; suporta múltiplos fabricantes; propõe uma arquitetura integrada de *software*; e utiliza políticas de gerenciamento. Percebe-se, entretanto, que tais características importantes foram pouco exploradas na literatura. Ao que tudo indica, o PipeConf é a primeira a apresentar todas as características no contexto da gerência de infraestrutura de rede.

3. Arquitetura PipeConf

A arquitetura proposta é genérica o suficiente para suportar diferentes políticas de fluxos de atividades. Contudo, essa flexibilidade traz desafios de pesquisa relacionados ao funcionamento eficiente de uma solução baseada na arquitetura PipeConf e à escalabilidade de tal solução. Esta seção descreve os detalhes da arquitetura, define três políticas de gerenciamento e apresenta os fluxos de trabalhos relacionados. Na seção seguinte, uma avaliação preliminar demonstra que a integração desenvolvida apresenta desempenho satisfatório.

3.1. Descrição da Arquitetura

A arquitetura de *software* integrado **PipeConf** provê a automatização da configuração de ativos de rede heterogêneos através da abordagem de infraestrutura como código. Ela emprega a estratégia de *pipeline* para garantir a modularidade e se baseia em políticas de fluxos de trabalho para permitir a adaptação automática do sistema. A Figura 1 apresenta a arquitetura proposta, que 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. Como indicado na figura, a comunicação entre estes módulos pode ocorrer a partir de diversos protocolos, a critério

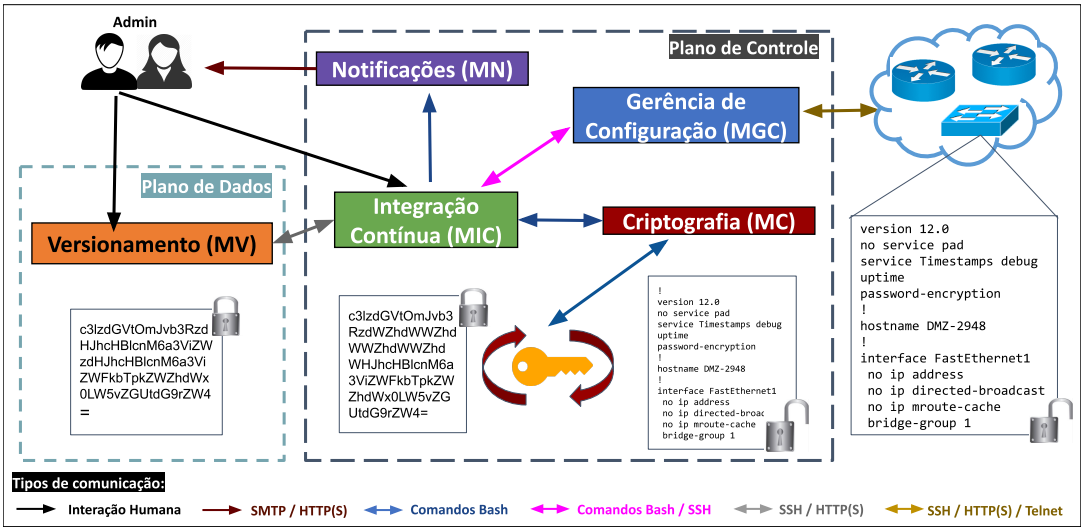


Figura 1. Arquitetura do PipeConf.

da equipe de administração da rede. Por exemplo, notificações podem ser feitas por email (SMTP) ou aplicações de colaboração (HTTPS); e a configuração dos ativos pode ser realizada por SSH, HTTP(S) ou Telnet. Internamente no Plano de Controle é possível ainda a comunicação via comandos *bash*. Pode ocorrer também por meio de SSH entre o MGC e o MIC, caso estes módulos sejam instanciados em computadores distintos.

O MV realiza o armazenamento e versionamento dos arquivos de credenciais e *backup* dos ativos de rede, para garantir não apenas a segurança no acesso, mas também a possibilidade de reversão em caso de problemas nas configurações dos mesmos. Para isso, uma ferramenta do tipo Git pode ser utilizada para armazenar os arquivos, que são previamente criptografados no Plano de Controle pelo MC. Este último também é responsável por decriptografar os arquivos quando necessário. O MGC é o módulo responsável por interagir diretamente com os ativos de rede e o MN notifica as pessoas interessadas em acompanhar os resultados das atividades. Por fim, o MIC interage com todos os módulos e executa o *pipeline* responsável por integrar as atividades de configuração dos ativos.

3.2. Políticas

O PipeConf emprega políticas para auxiliar no gerenciamento automático dos ativos de rede. Políticas permitem mudanças no comportamento de um sistema sem necessitar que o mesmo seja recompilado ou reinicializado. Neste trabalho, as políticas são descritas no formato "*SE(Condição) ENTÃO(Ação)*", onde a condição é o tipo de fluxo de trabalho a ser realizado e a ação são as atividades que o *pipeline* deve executar para completar tal fluxo. Foram então definidas três condições, que são acionadas quando: `ADD_ASSET` - o administrador da rede precisa cadastrar um ativo no PipeConf; `CONFIG_BACKUP` - uma cópia da configuração dos ativos é requisitada; ou `CONFIG_UPDATE` - uma alteração nas configurações dos ativos é necessária. Também foram definidas sete atividades que podem ser realizadas pelo PipeConf, que são: (i) **entrada** - interação humana ou agendada solicitando a cópia da configuração de ativos ou enviando arquivos contendo alterações a serem aplicadas; (ii) **criptografia** e (iii) **decriptografia** - os arquivos de credenciais e/ou configuração dos ativos são criptografados e decriptografados; (iv) **versionamento** - armazenamento e controle de versão dos arquivos; (v) **notificação** - envio de notificações

sobre o andamento das atividades; (vi) **cópia da configuração** - criação de arquivos contendo a configuração atual de cada ativo; e (vii) **alteração da configuração** - aplicação de alterações na configuração de um ou mais ativos.

A Figura 2 exibe as políticas utilizadas no PipeConf, definindo a sequência de atividades do *pipeline* a ser executado dependendo do fluxo de trabalho desejado. Durante a instanciação do *pipeline*, cada atividade pode ser executada por uma ferramenta específica de acordo com a implementação da arquitetura do PipeConf. Ressalta-se novamente que outras políticas podem ser criadas conforme novos fluxos de trabalho sejam adicionados.

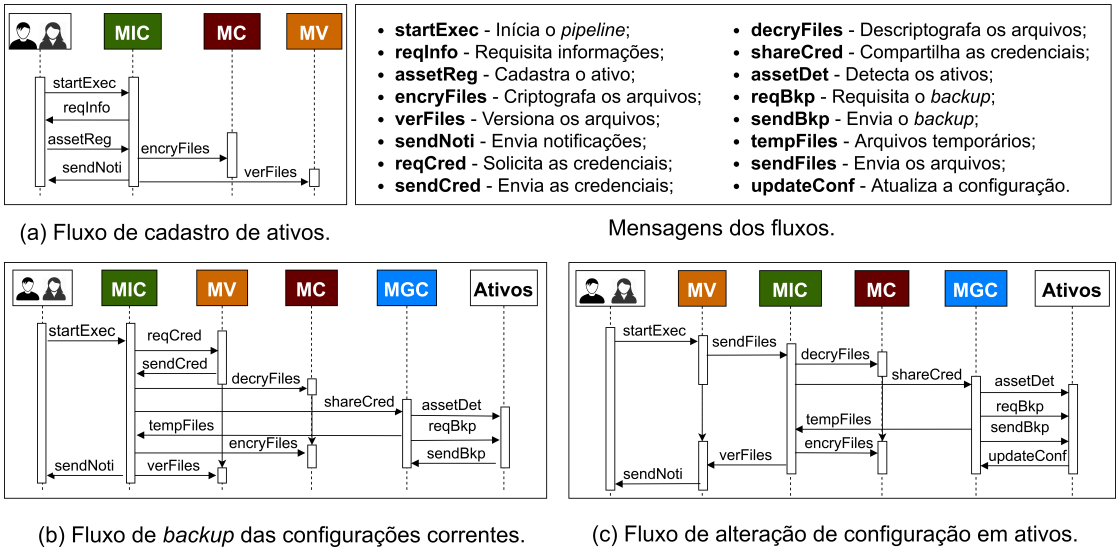
```
if fluxo_trabalho == ADD_ASSET
    then pipeline = entrada[criptografia;versionamento;
    notificação]
if fluxo_trabalho == CONFIG_BACKUP
    then pipeline = entrada[decriptografia;cópia da configuração;
    criptografia;versionamento;notificação]
if fluxo_trabalho == CONFIG_UPDATE
    then pipeline = entrada[decriptografia;cópia da configuração;
    alteração da configuração;cópia da configuração;criptografia;
    versionamento;notificação]
```

Figura 2. Políticas dos fluxos de trabalho

3.3. Fluxos de Trabalho

Conforme os fluxos de trabalho expostos na Figura 3, o PipeConf pode ser utilizado para cadastrar as credenciais de acesso, obter o *backup* e/ou aplicar alterações na configuração de ativos. A Figura 3(a) apresenta o fluxo das atividades realizadas quando o administrador precisa cadastrar as credenciais de acesso aos ativos. Essas credenciais são criptografadas antes de serem armazenadas em um sistema de controle de versão. A Figura 3(b) apresenta o fluxo quando é necessário fazer apenas o *backup* da configuração dos ativos na rede. Os arquivos gerados contêm a sintaxe nativa de comandos de cada modelo e fabricante e a configuração é tratada como código, sendo versionada e possibilitando o rastreamento de mudanças e reversão em caso de problemas. A Figura 3(c) mostra o fluxo para alterar a configuração de um ou mais ativos de rede. Nesse fluxo ocorre uma dupla execução de *backup* em cada ativo, antes e após as alterações das configurações, possibilitando assim a reversão (do inglês *rollback*) da configuração em caso de problemas. As notificações podem ser enviadas pelo PipeConf por email ou aplicativo de colaboração (p.ex. Slack), contendo informações sobre a execução das atividades. Diversos tipos de notificações podem ser implementados, a depender de como a arquitetura é instanciada.

Em todos os fluxos apresentados na Figura 3 são exibidas mensagens que indicam: `startExec` - o início da execução do *pipeline* após uma interação humana ou agendada; `reqInfo` - a solicitação das informações a cerca do modelo de ativo a ser cadastrado; `assetReg` - o cadastro do ativo de rede; `encryFiles` - a criptografia do arquivo contendo as credenciais e/ou *backup* dos ativos; `verFiles` - o versionamento dos arquivos; `sendNoti` - o envio de notificações; `reqCred` - a solicitação das credenciais de acesso aos ativos; `sendCred` - o envio das credenciais dos ativos; `decryFiles` - a descriptografia dos arquivos de credenciais e/ou configuração dos ativos; `shareCred` - o compartilhamento das credenciais de acesso; `assetDet` - a detecção dos ativos conectados; `reqBkp` - a criação do *backup* da configuração dos ativos; `sendBkp` - o envio do *backup* da configuração; `tempFiles` - o armazenamento temporário dos arquivos de *backup*; `sendFiles` - o envio das credenciais e arquivos com a nova configuração dos ativos; e `updateConf` - a alteração da configuração dos ativos.



(a) Fluxo de cadastro de ativos.

Mensagens dos fluxos.

(b) Fluxo de *backup* das configurações correntes.

(c) Fluxo de alteração de configuração em ativos.

Figura 3. Fluxos de trabalho.

4. Avaliação

Esta seção apresenta uma avaliação da arquitetura PipeConf através de uma **Prova de Conceito** (PoC, do inglês *Proof of Concept*), instanciada a partir de ferramentas consolidadas em ambientes de IaC. A PoC consiste em gerenciar dispositivos simulados de rede, buscando alterar a configuração corrente desses ativos, realizar *backup* dos arquivos e versioná-los em um repositório de código. O objetivo é validar a capacidade do PipeConf em gerenciar ativos de maneira eficiente; ou seja, executar atividades em um número crescente de ativos, sem aumentar excessivamente o tempo de execução das mesmas. Outro fator importante associado à eficiência é a gerência de dispositivos heterogêneos.

O código fonte e a documentação de uso do PipeConf estão publicados em um repositório de livre acesso[1]. As ferramentas a seguir foram selecionadas pelos recursos mínimos exigidos de CPU e memória, além das funcionalidades requeridas em cada módulo do PipeConf. São elas: **Gogs** e **PostgreSQL** (ambas situadas no MV); **Jenkins** (localizada no MIC); **Sops** (situada no MC); **Salt SProxy** e **Napalm** (ambas localizadas no MGC); **Postfix** (localizada no MN); e **Docker** (para portabilidade entre distribuições Linux). Detalhes dessas ferramentas estão descritos na documentação do repositório. Embora sejam utilizadas separadamente e em diferentes contextos, a arquitetura PipeConf realiza a integração das mesmas por meio de um *pipeline* escrito como código, visando gerenciar a configuração automatizada de *switches* e roteadores. Importante ressaltar, contudo, que a arquitetura modular do PipeConf permite o uso de soluções alternativas.

Os experimentos descritos na Subseção 4.1 buscam analisar o comportamento do PipeConf mediante fatores pré-definidos de entrada, levando em consideração os aspectos de quão rápida é a execução e o impacto da quantidade de ativos. As Subseções seguintes estão relacionadas com os seguintes objetivos específicos dessa avaliação: i) (4.2) Perfilar o tempo de execução de cada atividade realizada pelo PipeConf e identificar possíveis pontos de "gargalo"; ii) (4.2) Comparar o tempo de execução de atividades realizadas pelo PipeConf e por uma ferramenta existente no mercado, que também ofereça suporte a ativos de rede heterogêneos; iii) (4.3) Analisar o desempenho do PipeConf ao realizar

[1]https://gitlab.com/aeciopires/pipeconf

atividades em uma quantidade significativa de ativos, bem como analisar o consumo de recursos como CPU e memória; iv) (4.4) Demonstrar não apenas que o PipeConf é capaz de administrar ativos heterogêneos, mas também discutir a hipótese de que a sua utilização normaliza os tempos de gerência de modelos distintos.

4.1. Ambiente de Experimentação

O ambiente de experimentação consiste em ativos de rede simulados no GNS3[2], tendo em vista que a utilização de ativos físicos reais requer um espaço físico apropriado para as instalações e também um alto investimento financeiro. Dois modelos foram selecionados para as simulações por critérios de tamanho da imagem simulada e sintaxe de comandos conhecida, são eles o roteador Cisco modelo C3640-IK9O3S-M e o *switch* Arista VeoS.

O ambiente computacional para execução dos experimentos consistiu de um *laptop* para execução do PipeConf e um servidor na GCP (*Google Cloud Platform*) para execução do GNS3, com as seguintes configurações: (*laptop*) CPU i5-4210U 1.70GHz e 4 núcleos, 8GB de RAM, Ubuntu Desktop 18.04 e SSD SATA III 480GB; e (servidor) CPU Intel(R) Xeon(R) 2.0GHz e 24 núcleos, 40GB de RAM, Ubuntu Server 18.04 e SSD SATA III 30GB. As versões dos *softwares* utilizados e os procedimentos de instalação do ambiente de experimentação estão disponíveis na documentação do PipeConf.

Foi definida uma sequência de atividades do PipeConf em consonância com o fluxo de trabalho de alteração da configuração de ativos (Figura 3(c)), a partir de uma configuração genérica (configurar o *Network Time Protocol*) que pode ser realizada em ativos de diferentes modelos e fabricantes. Assim, os experimentos podem ser facilmente replicados e o que realmente influencia nos resultados é a quantidade de ativos de rede. A principal métrica é o **tempo médio de execução** da sequência de atividades descritas na Tabela 1. Apesar de ser considerada uma atividade genérica na **AT07**, ressalta-se que qualquer configuração é possível nos ativos gerenciados.

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

Tabela 1. Descrição das atividades executadas pelo PipeConf.

Nas subseções seguintes, as análises realizadas baseiam-se em medir o tempo médio de execução do PipeConf à medida que o número de ativos configurados dobra. O tempo de execução foi computado a partir do início da primeira atividade (**AT01**) até a conclusão da última (**AT12**). Cada experimento foi executado 30 vezes para prover uma confiabilidade estatística mínima, com um nível de confiança de 95%, e com intervalos de

[2]https://gns3.com

confiança apresentados sempre em relação à média. O consumo dos recursos do sistema foi coletado por meio de medições no computador onde o PipeConf foi instalado. A coleta das métricas de consumo de recursos ocorreu de forma paralela às atividades executadas pelo PipeConf, com intervalos de 60 segundos entre as mesmas.

Ativos de redes geralmente necessitam de uma configuração prévia para estarem acessíveis na rede, por exemplo, endereçamento IP. Assim, os tempos para instalação física e configuração inicial dos ativos não foram considerados nos resultados. Portanto, pressupõe-se que cada ativo já está devidamente inicializado e acessível. Além disso, por questões de economia de espaço, as duas subseções seguintes relatam os resultados obtidos apenas com o modelo Cisco de ativo simulado. Por fim, são apresentados os resultados envolvendo o *switch* Arista e uma discussão sobre a heterogeneidade dos ativos.

4.2. Análise de Perfilamento

Os principais propósitos desta análise são: (i) caracterizar os tempos de execução das atividades do PipeConf e (ii) analisar comparativamente com outra solução existente no mercado. Além disso, com o perfilamento proposto foi possível identificar pontos de melhoria no sistema, a partir do tempo de execução de algumas atividades. A análise comparativa também possibilitou detectar um "gargalo" na execução paralela do *pipeline* devido a uma ferramenta específica, conforme detalhado mais adiante. Contudo, encontrar uma solução alternativa para comparar com o PipeConf não foi uma tarefa trivial.

Resumidamente, não foi encontrada uma solução (de mercado ou acadêmica) que realize todas as atividades executadas pelo PipeConf. Além disso, fatores como a ausência de licenças gratuitas, o curto tempo de avaliação e difícil usabilidade se mostraram impeditivos em grande parte das 10 soluções investigadas[1]. Destas, apenas o **Unimus**[3] se mostrou viável, tendo em vista que a equipe de desenvolvedores gentilmente disponibilizou, para fins desta pesquisa, uma licença temporária para a configuração de até 32 ativos simultâneos (a licença padrão permite apenas 5). Apesar do Unimus executar apenas 2 das 12 atividades descritas na subseção anterior, ainda assim optou-se por comparar as duas soluções com o intuito de apresentar mais evidências da eficiência do PipeConf. As atividades realizadas em comum são: **AT05** - Fazer *backup* da memória (*running-config*) antes da alteração; e **AT07** - Alterar configuração do servidor NTP (a.ntp.br). Os procedimentos de instalação também estão disponíveis na documentação do PipeConf.

Inicialmente foram perfilados os tempos de execução de cada atividade executada pelo PipeConf com até 32 ativos simultâneos, para efeito de posterior comparação com o Unimus. A Figura 4 apresenta a distribuição dos tempos médios (em minutos) para executar cada uma das 12 atividades, levando em consideração 30 repetições de cada experimento. Um importante destaque é o tempo médio total de configuração relativamente baixo (menos de 10 minutos para configurar 32 dispositivos simultâneos). Percebe-se também como cada atividade impacta no tempo médio total de configuração e como este impacto aumenta com o número de ativos. Por exemplo, verifica-se que a **AT01** influencia pouco no tempo total independente do número de ativos, já que o acesso ao repositório Git ocorre sempre muito rápido. Já a decriptografia da **AT02** demora aproximadamente 2 minutos para ser executada e apresenta um comportamento de tempo constante devido à forma como os arquivos de credenciais são identificados dinamicamente.

[3]https://unimus.net

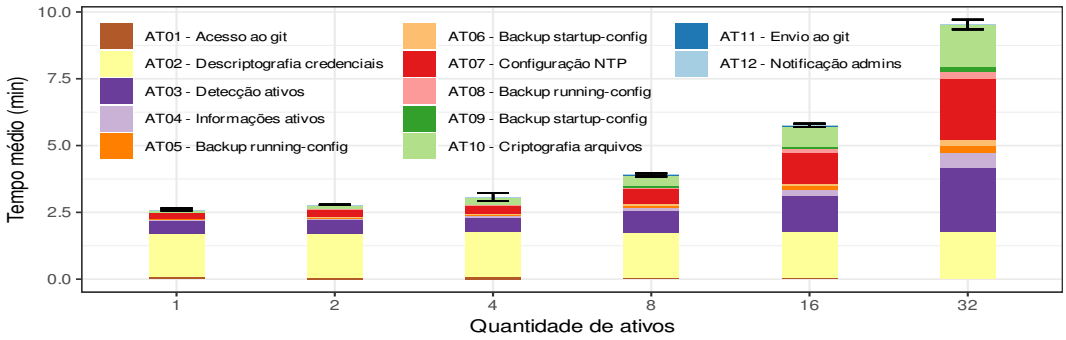


Figura 4. Perfilamento dos tempos médios de configuração em cada atividade.

Observa-se ainda que o tempo da **AT03** cresce com o aumento de ativos, pois um contêiner do Salt SProxy é iniciado e executa uma função interna do código SaltStack para detectar quais estão acessíveis. Essa verificação não é uma simples troca de mensagem, mas sim a execução do módulo *net.connected*[4], que demora alguns segundos. Ao final, o contêiner do Salt SProxy é interrompido. No *log* do PipeConf é possível constatar quais ativos estão ou não acessíveis durante as atividades. Na **AT04**, outro contêiner do Salt SProxy é iniciado e continua executando até o fim da **AT09**. Neste ínterim, o PipeConf obtém algumas informações de cada ativo, tais como: fabricante, modelo, sistema operacional, arquitetura computacional, entre outras. Essas informações são necessárias ao Napalm (biblioteca usada pelo Salt SProxy), para determinar qual sintaxe de comandos será utilizada para interagir com o ativo antes de iniciar as atividades seguintes.

As atividades **AT05** e **AT06** realizam o *backup* das configurações antes de realizar alterações, o que possibilita reversão das ações caso aconteça algum problema. No geral demoram poucos segundos, mas tendem a aumentar o tempo com o número de ativos. A **AT07** mostrou-se impactar predominantemente na configuração de um grande número de ativos. Isto ocorre porque as configurações são paralelizadas de acordo com o número de núcleos de processamento disponíveis, conforme detalhado mais adiante. As atividades **AT08** e **AT09** comportam-se de maneira semelhante às **AT05** e **AT06**, respectivamente. Ao final da **AT09**, o contêiner do Salt SProxy é encerrado e recursos de CPU e memória são liberados. O tempo da **AT10** cresce com o aumento de ativos e testar ferramentas alternativas de criptografia é certamente uma perspectiva de trabalho futuro. Por fim, **AT11** e **AT12** são rápidas e variam pouco em relação às demais atividades.

Esta análise de perfilamento foi importante pois possibilitou detectar que o crescimento no tempo de execução da **AT07** (configurar NTP) está relacionado ao número de núcleos de processamento presentes no *laptop* utilizado[5]. Mais especificamente, o Salt SProxy inicializa a configuração de até 4 ativos simultâneos por vez, tendo em vista que tal *laptop* possui apenas 4 núcleos. Dessa forma, um maior benefício de realizar mais atividades em paralelo não é alcançado. Este comportamento fica ainda mais evidente em uma comparação entre o PipeConf e o Unimus, retratada na Figura 5 e na Tabela 2.

Percebe-se que o Unimus apresenta tempos de execução das atividades bem mais estáveis com o aumento no número de ativos (Figura 5(a)). Já com o PipeConf (Figura 5(b)), o aumento nos tempos das atividades equivalentes é bem acentuado com mais ati-

[4]https://bit.ly/2MOQrWA

[5]https://bit.ly/3bAkQ5b

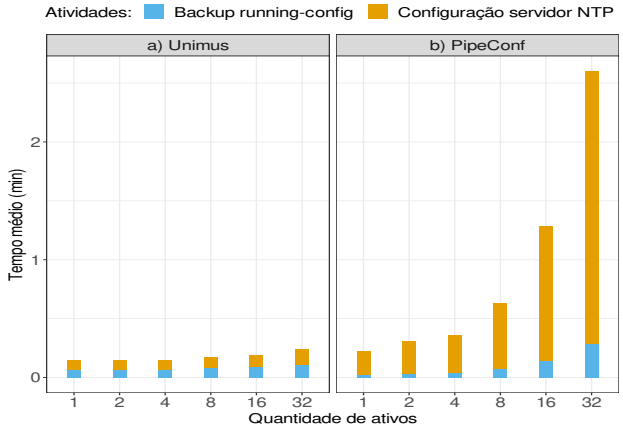


Figura 5. Perfilamento das atividades comuns entre Unimus e PipeConf.

Nº de Ativos	Unimus (erro máximo 0.01)	
	Backing up	Configurando
1	0, 07	0, 08
2	0, 07	0, 08
4	0, 07	0, 08
8	0, 08	0, 09
16	0, 09	0, 1
32	0, 11	0, 13
Nº de Ativos	**PipeConf (erro máximo 0.07)**	
	Backing up	Configurando
1	0, 02	0, 2
2	0, 03	0, 28
4	0, 04	0, 32
8	0, 07	0, 56
16	0, 14	1, 14
32	0, 29	2, 31

Tabela 2. Valores dos tempos médios em minutos.

vos, principalmente devido à limitação das tarefas simultâneas na **AT07**. Com base em uma avaliação empírica, o Unimus inicializa um processo paralelo para configuração de cada ativo gerenciado e deixa o sistema operacional responsável por gerenciar tais processos. O ganho de desempenho é significativo, principalmente a partir de 16 ativos.

A Tabela 2 apresenta os valores absolutos dos tempos médios das atividades comuns (verde significa melhores tempos). Destaca-se que para realizar o *backup* das configurações o tempo não é tão discrepante entre o Unimus e PipeConf, chegando a ser melhor com o último para até 8 ativos e com uma diferença máxima de aproximadamente 10 segundos (0,18 minuto) para 32. Por outro lado, para configurar o servidor NTP (**AT07**), os tempos com o Unimus são sempre menores, novamente, por usufruir de um maior paralelismo na inicialização desta atividade.

4.3. Análise de Desempenho

Esta análise apresenta os resultados de desempenho do PipeConf em relação ao tempo médio de configuração de até 128 ativos simultâneos (limite máximo simulado no GNS3 com a imagem do modelo Cisco). Adicionalmente, foram coletadas métricas de consumo dos recursos computacionais exigidos pelo PipeConf durante os experimentos.

A partir da análise de perfilamento da **AT07** (configurar NTP) do PipeConf, esta análise busca também demonstrar o benefício de um maior nível de paralelismo a partir de um novo conjunto de experimentos. A ideia é atestar que configurar mais ativos simultaneamente repercute no tempo total obtido. Para tal, repetiu-se os mesmos experimentos até 128 ativos, porém com o Salt SProxy executado no servidor da GCP que possui 24 núcleos de processamento. Esta nova configuração foi denominada de **PipeConf++** e os resultados das duas versões dos experimentos estão na Figura 6. Como esperado, o tempo médio gasto pelo PipeConf para realizar as atividades aumenta com o número de dispositivos configurados. Contudo, este aumento não segue a mesma dimensão do acréscimo na quantidade de ativos, indicando um ganho proporcional à medida que mais dispositivos são configurados. O tempo médio de configurar 128 ativos (58,13 minutos), por exemplo, não é 128 vezes maior do que configurar um único ativo. Na realidade, há um ganho proporcional de aproximadamente 83% em relação ao tempo médio de configurar um único ativo (2,61 minutos). Este comportamento é intuitivo e se deve ao fato do PipeConf

conseguir executar algumas tarefas em paralelo, embora de maneira limitada.

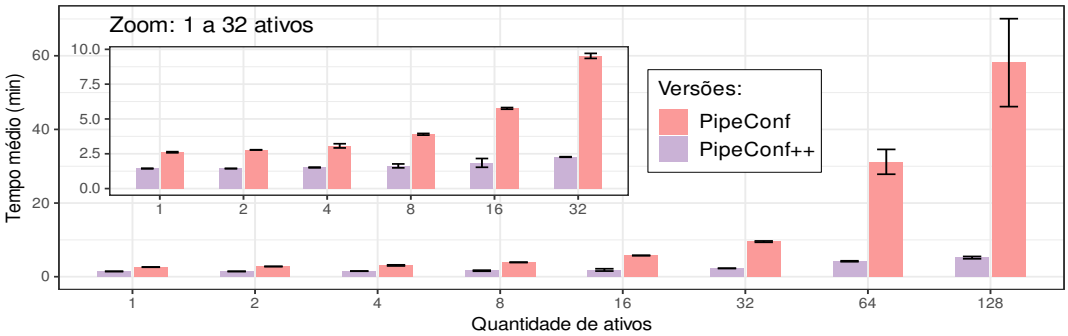


Figura 6. Tempos médios de configuração para as duas versões do PipeConf.

A figura demonstra claramente o benefício de paralelizar as atividades de configuração do PipeConf, ainda mais evidente com um grande número de ativos gerenciados. O PipeConf tende a dobrar o tempo de execução à medida que dobra o número de ativos de rede gerenciados, levando em média aproximadamente 58 minutos para gerenciar 128 ativos. Por outro lado, o PipeConf++ apresenta uma taxa de crescimento bem menor do que os tempos do PipeConf. Para gerenciar os mesmos 128 ativos, por exemplo, o PipeConf++ leva em média 5,18 minutos. Entretanto, ressalta-se que o objetivo não é comparar diretamente as duas versões apresentadas do PipeConf, tendo em vista que foram executadas em máquinas distintas. O objetivo é demonstrar o potencial benefício de iniciar mais configurações simultâneas pelo mesmo sistema. O entendimento de como essas tarefas podem ser paralelizadas também é importante para melhoria do PipeConf.

Para mostrar a escalabilidade do PipeConf, foram coletadas métricas de utilização dos recursos computacionais do *laptop* onde os componentes de *software* foram executados (vide Seção 4.1), ao longo de todos os experimentos realizados neste trabalho. A Figura 7 apresenta quatro destas métricas, que foram monitoradas a cada 60s de um total de aproximadamente 77h de execução. A figura exibe a carga no sistema (azul); os consumos de CPU e memória (respectivamente verde e mostarda); e a quantidade de contêineres em execução no sistema (rosa), ao longo de todo o período de experimentação. A linha tracejada nos gráficos indica a média móvel da respectiva métrica a cada hora.

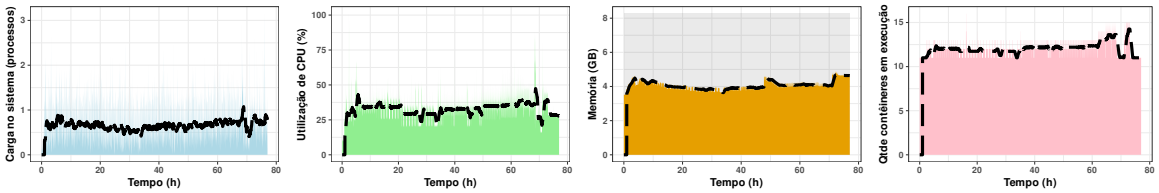


Figura 7. Métricas do consumo de recursos pelo PipeConf nos experimentos.

De uma maneira geral, a Figura 7 demonstra que o consumo dos recursos permanece estável ao longo dos experimentos, independentemente da quantidade de ativos configurados pelo PipeConf. A carga no sistema (azul) representa a quantidade de processos nos estados de execução ou espera, cuja média é predominantemente abaixo de uma unidade e que significa menos de um núcleo ocupado 100% do tempo. Um regra geral assumida para sistemas estáveis é que a carga média no sistema não deve exceder o número de núcleos lógicos presente no computador[6]. Quanto ao consumo de CPU (verde)

[6]https://bit.ly/3qFCSqY

e memória (mostarda), observa-se também uma regularidade no uso destes recursos, com médias quase sempre inferiores à metade do total disponível e poucos picos esporádicos ao longo do tempo. Por fim, o número de contêineres Docker (rosa) utilizados na operação do PipeConf variou em torno de 12 unidades. Este inicia sob demanda um contêiner para executar o Salt SProxy, que realiza das atividades **AT03** à **AT09**. Ao final desta última, o contêiner é encerrado e recursos de CPU e memória são liberados. Esse ciclo acontece durante cada uma das 30 repetições, independente do número de ativos. Destaca-se portanto que, mesmo com o número crescente de ativos gerenciados simultaneamente, a quantidade de contêineres no sistema continuou estável a partir do reaproveitamento de recursos, o que indica uma boa eficiência do PipeConf no uso dos mesmos.

4.4. Análise da Heterogeneidade

Um premissa importante do PipeConf é a interoperabilidade de modelos distintos, o que atenua a necessidade de aprender novas sintaxes de comandos por parte dos administradores da rede. Conjectura-se ainda que, com a utilização da solução proposta, é possível gerenciar ativos com sintaxes diferentes em um período de tempo similar. A configuração manual ou mesmo através de ferramentas separadas que não executam um processo holístico é provavelmente mais demorada, além de ocasionar inconsistências e falhas na rede. Nesse contexto, também foram realizados experimentos com um modelo do *switch* Arista VeoS, cujo tamanho da imagem possibilitou simular apenas 16 ativos simultâneos. Comparativamente, no mesmo servidor com 40GB de RAM foram simulados até 128 roteadores Cisco modelo C3640-IK9O3S-M. Portanto, as Figuras 8 e 9 apresentam uma análise preliminar sobre a heterogeneidade de ativos.

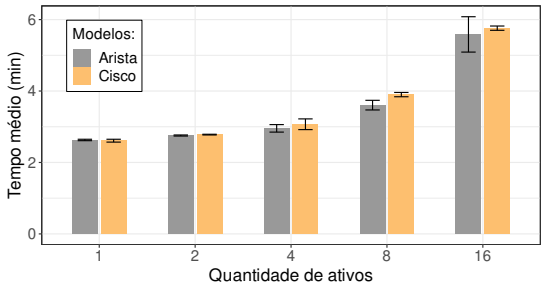


Figura 8. Comparação dos tempos médios de configuração.

Figura 9. Diferença média entre os tempos dos dois modelos.

A Figura 8 apresenta uma comparação do tempo médio de configuração das 12 atividades descritas anteriormente, levando em consideração os dois modelos distintos de ativos. Observa-se que os tempos são muito similares, com uma ligeira diferenciação a partir de 4 ativos. Aparentemente, a configuração com o Arista tende a ser um pouco mais rápida, porém com maior variabilidade. Contudo, a interseção dos intervalos de confiança não permite uma distinção precisa de tais tempos e, para tanto, a Figura 9 apresenta os resultados a partir de *testes-t* [Jain 1991] para a hipótese de dados similares. Pela figura percebe-se que os intervalos de confiança das diferenças entre as amostras dos dois modelos interceptam o valor 0 para 1, 4 e 16 ativos (com valor de $p > 0.05$), o que significa estatisticamente que o tempo de configurar com o Arista não é diferente do Cisco. Por outro lado, para 2 e 8 ativos simultâneos, não se pode afirmar tal indiferença, mesmo com valores médios muito próximos de 0. Resumidamente, os resultados mostram indícios de que os tempos de configuração com ativos distintos são similares, mas uma análise mais detalhada se faz necessária para comprovar definitivamente tal hipótese.

5. Conclusão

Este trabalho propôs uma arquitetura integrada de *software* que utiliza a abordagem de Infraestrutura como Código para gerenciar a configuração de ativos de rede heterogêneos. Uma prova de conceito foi desenvolvida em um ambiente simulado, a fim de realizar uma avaliação quantitativa do tempo de execução e escalabilidade do sistema. Foram realizados experimentos variando o número de ativos simultâneos e coletadas diversas métricas sobre a execução do PipeConf. Os resultados obtidos demonstraram um desempenho satisfatório, mesmo diante de um grande número de ativos gerenciados. Como trabalhos futuros, será investigado um outro componente de *software* que paralelize o gerenciamento dos ativos independente do número de núcleos físicos de processamento, além de uma alternativa para criptografar arquivos de maneira mais rápida.

Referências

Chen, H. et al. (2018). Towards Example-Guided Network Synthesis. In *Proceedings of the ACM 2nd Asia-Pacific Workshop on Networking*, APNet '18, pages 65–71, New York, NY, USA.

Christopher, R. (2019). Strategic IT management: how companies can benefit from an increasing IT influence. *J. of Enterprise Information Management*, 32(2):251–273.

Cox, J. H. et al. (2017). Advancing Software-Defined Networks: A Survey. *IEEE Access*, 5:25487–25526.

Duplyakin, D. et al. (2015). Architecting a Persistent and Reliable Configuration Management System. In *Proc. of the ACM 6th Workshop on Scientific Cloud Computing*, pages 11–16.

Ismail, H. et al. (2018). Semantic Enhancement for Network Configuration Management. In *2018 IEEE Global Conference on Internet of Things (GCIoT)*, pages 1–5.

Jain, R. (1991). *The art of computer systems performance analysis - techniques for experimental design, measurement, simulation, and modeling.* Wiley.

Jiang, Y. and Adams, B. (2015). Co-Evolution of Infrastructure and Source Code: An Empirical Study. In *Proc. of the 12th Working Conference on Mining Software Repositories*, pages 45–55.

Liu, H. H. et al. (2018). Automatic life cycle management of network configurations. In *Proc. of the Afternoon Workshop on Self-Driving Networks*, SelfDN'18, page 29–35, NY, USA.

Morris, K. (2016). *Infrastructure as Code - Managing Servers in the Cloud.* O'Reilly Media, Inc., Sebastopol, CA, USA, 1st edition.

Netto, H. V. et al. (2017). Coordenação de Containers no Kubernetes: Uma Abordagem Baseada em Serviço. In *Anais do 35º SBRC*, Belém, PA, Brasil. SBC.

Oliveira, W. F. et al. (2018). Aprovisionamento de QoS por Rótulos Programáveis para Redes Definidas por Resíduos. In *Anais do 36º SBRC*, Campos do Jordão, SP, Brasil. SBC.

Opara-Martins, J. et al. (2016). Critical analysis of vendor lock-in and its impact on cloud computing migration: a business perspective. *Journal of Cloud Computing*, 5(1):4.

Sette, I. et al. (2019). Mapeamentos Sintático e Semântico em Federações de Políticas de Autorização para Nuvens Heterogêneas. In *Anais do 37º SBRC*, Gramado, RS, Brasil. SBC.

Shah, J. A. and Dubaria, D. (2019). NetDevOps: A New Era Towards Networking & DevOps. In *IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference*.

Vilalta, R. et al. (2020). Network Programmability and Automation in Optical Networks. In *Optical Network Design and Modeling*, Cham. Springer International Publishing.