設定ファイル作成の自動化によるNginxサーバ構築の簡略化

栗原 尚希[1]　大野 有樹[2]　串田 高幸[1]

概要：Web サーバのシェアの拡大に伴い，Nginx のシェアも増加している．Nginx を Web サーバとして構築する際には，設定ファイルへの設定項目の記述作業が必要となる．しかし，設定項目は無数に存在するため必要とする設定に対応する項目を探して設定ファイルに正確に記述するには時間を要する．課題はリバースプロキシを構築する際の所要時間である．本稿の提案手法は，k3s のインストールから Pod 生成までの自動化である．課題を立証する実験結果として，リバースプロキシ構築未経験者が手作業で構築を行った場合において，作業完了までに 225 分の時間が掛かった．

1. はじめに

背景

Nginx と呼ばれる Web サーバのシェアが増加している [1, 2]．Web サーバはユーザからのリクエストを受けて処理を実行し，ユーザにレスポンスを返すためのコンピュータである*1．Nginx は Web サーバソフトの中でも，処理性能の高さを売りに開発されていることから，同時に複数の処理を高速で実行できる．さらに他の特徴としてリバースプロキシ機能を有していることが挙げられる*2．

リバースプロキシはクライアントと Web サーバの通信の間に入って，Web サーバの応答を代理しつつ通信を中継する機能である．プロキシとリバースプロキシの違いとして，プロキシはクライアント側の Web ブラウザが Web サーバに送るリクエストを代理する．クライアント側の匿名性確保，コンテンツ表示の高速化やアクセスログの確保が役割として挙げられる [3, 4]．一方でリバースプロキシは Web サーバ側のクライアントへの応答を代理し，セキュリティ対策や性能向上，負荷分散，システム構成の自由度向上のために利用される*3．

本稿では Nginx を直接仮想マシン (VM) にインストールせず，コンテナ上で運用することを前提としている．なぜならコンテナ技術は VM に比べ，起動や管理に必要とするリソースのオーバーヘッドが少なく軽量で高速かつ，移植性が高い仮想化技術である．さらにコンテナはアプリケーションとその依存関係をパッケージ化し，異なる環境でも一貫した動作をさせることができる [5]．そのためアジャイル開発によるサービスレベルの向上や DX の推進を目的とする企業で利用されている*4．

一方でコンテナにはアプリケーションを実行する機能はあるが，コンテナを管理したり他のサーバと連携させる機能はない．この問題を解決するための，Kubernetes と呼ばれるオープンソースソフトウェアが存在する*5．Kubernetes はコンテナ化されたアプリケーションを管理し，デプロイ，スケーリング，及び運用するためのコンテナオーケストレーションツールのひとつである [6]．本稿では Kubernetes の代わりに k3s を使用する．k3s は Kubernetes の機能をスリム化することで，小規模なコンピュータでも実行可能にしたものである．

コンテナを作成する際，ConfigMap と呼ばれる Kubernetes のリソースを使用することができる．ConfigMap はアプリケーションの構成情報を保存し，それをコンテナ内のアプリケーションに提供するためのリソースである．本稿では Nginx の設定ファイルの nginx.conf の構成情報を ConfigMap に保存する [7, 8]．

課題

本稿では Web サーバの高速化のためのリバースプロキシ構築を行うことを前提としている．図 1 はリバースプロキシやクライアントの位置関係を示した経路図である．

Nginx を Web サーバとして構築する際には，設定ファイルの記述作業が必要である．この作業は主に設定ファイル内の設定項目の記述を行う．ユーザが要求する機能によっ

[1] 東京工科大学コンピュータサイエンス学部
〒 192–0982 東京都八王子市片倉町 1404-1
[2] 東京工科大学院バイオ・情報メディア研究科
〒 192–0982 東京都八王子市片倉町 1404-1

*1 https://cn.teldevice.co.jp/column/38275/
*2 https://wa3.i-3-i.info/index.html
*3 https://atmarkit.itmedia.co.jp/ait/articles/1608/25/news034.html
*4 https://atmarkit.itmedia.co.jp/ait/articles/2107/30/news032.html
*5 https://udemy.benesse.co.jp/development/system/kubernetes.html

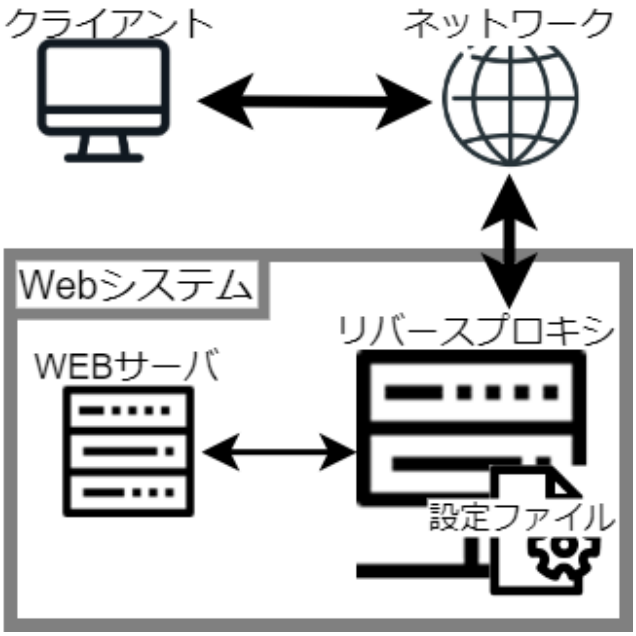

図 1　リバースプロキシの説明

て記述必要な設定項目が違い，例としてリバースプロキシのサーバを構築する際に必要な設定項目を以下に挙げる．

- ”server”ブロック:
  特定のポートやドメインに対するサーバの設定を定義する．
- ”listen”ディレクティブ:
  Nginx がリクエストを受け付けるためのポート番号や IP アドレスを指定する．
- ”location”ブロック:
  リクエストのパスに基づいて Web サーバにプロキシするための設定を定義する．
- ”proxy_pass”ディレクティブ:
  プロキシ先の Web サーバのアドレスを指定する．
- ”proxy_set_header”ディレクティブ:
  リクエストヘッダの情報を変更または追加を行う．

このようにリバースプロキシとして機能させるだけでも，記述が必要な設定項目は複数ある．これ以外にも，リクエストヘッダーの情報を Web サーバに転送するための”proxy_set_header”ディレクティブや，リバースプロキシのサーバと Web サーバがサーバ間でセキュアな通信を確立するための”proxy_ssl”ディレクティブの，無数の設定項目が存在している．ソースコード 1 は nginx.conf ファイルのサンプルである．Nginx を正常に動作させるには，設定ファイルにこれらの項目をソースコード 1 のように，Nginx 独自のドキュメントルールに則って正しく記述する必要性がある．Web システムの構築経験の無いサーバ初学者がドキュメントルールを確認しつつ，必要とする設定に対応する項目を公式ドキュメント[*6]から探し出し，設定ファイルを作成する場合，Web システムの熟練者に比べ時間を要することとなる．本稿ではこの所要時間を課題とする．図 2 は課題を表したものである．

[*6] http://nginx.org/en/docs/

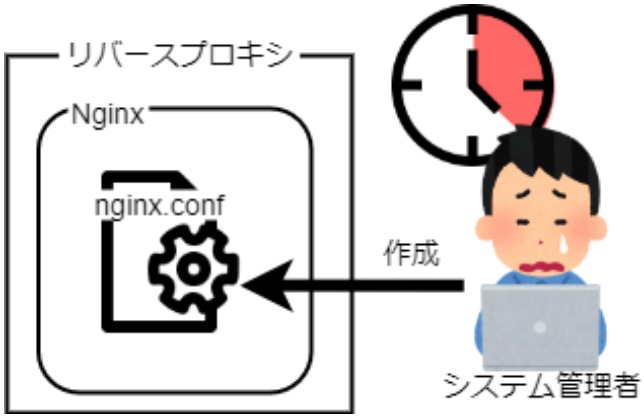

図 2　設定ファイル作成に時間を要す

ソースコード 1　nginx.conf のサンプル

```
1  worker_processes 1;
2  events {
3      worker_connections 1024;
4  }
5  http {
6      server {
7          listen 80;
8           server_name localhost;
9
10         location / {
11             root html;
12             index index.html index.html;
13          }
14
15         error_page 500 502 503 504 /50x.html;
16          location = /50x.html {
17             root html;
18          }
19     }
20 }
```

各章の概要

本稿は以下のように構成されている．第 1 章では背景と課題について述べる．第 2 章では関連研究について述べる．第 3 章では本稿での課題解決するための提案方式を説明する. 第 4 章では実装及び実験方法を述べる. 第 5 章では基礎実験の内容と，その評価について述べる. 第 6 章では本稿の提案方式についての議論を行う. 第 7 章では本稿のまとめと成果を述べる.

2. 関連研究

ネットワーク構成の設定を自動化する必要性を指摘している論文がある [9]．この論文では自動化はネットワーク構成の問題の解決策として複数の論文で提案されてきたとしている．本稿も自動化を提案し Nginx サーバの構築を簡略化することでネットワーク構成の設定を自動化できると言える．

Nginx ベースの Web サーバのチューニングを行っている研究がある [10]．これは Web サーバのスレッドやプロセスの値，キャッシュサイズの設定を，最適化・分析をしている．これにより，Web サーバの総合的なパフォーマンスが向上した．本稿の想定するサーバ環境も，この研究同様に Nginx ベースである．本提案ソフトウェアで構築されるサーバは最適化はされておらず，行った場合にサーバのパフォーマンス向上が期待できる．本提案ソフトウェアにチューニング機能を追加する場合，この研究を参考にできる．

マイクロサービスベースのアプリケーションのアーキテクチャを自動的に決定する研究がある [11]．この手法では 3 段階のマイニングを行うことで，アプリケーションのアーキテクチャのトポロジーを理解することができる．しかし，リバースプロキシとして Nginx をデプロイするために必要な記述を ConfigMap に自動的に行うことはできない．本稿では，ConfigMap に各サービスの IP アドレスを自動的に記述する手法を提案する．

3. 提案

提案方式

本稿では，Nginx で構成されたリバースプロキシの構築時間を削減するために，リバースプロキシ構築作業の自動化を提案する．本稿での提案するソフトウェアの流れを以下の図 3 に示す．

図 3 提案ソフトウェアの流れ

図 3 に示した通り，提案ソフトウェアは 4 つの工程に分けられる．システム管理者がリバースプロキシとしたいサーバで提案ソフトウェアを実行後，工程①として k3s のインストールを行う．本提案は，新たにリバースプロキシ用のサーバを用意しただけの状況を前提としており，コンテナ管理するために”kubectl”コマンドを使用可能な状態にする必要がある．

図 3 の②では，プロキシ先サーバの探索を行う．サーバをリバースプロキシとして機能させるには，プロキシ先の情報がリバースプロキシの設定ファイルに記されている必要がある．②はプロキシ先の情報を得るための工程であり，詳細を図 4 に示す．探索は図 4 に示した通り，KubernetesCluster 内かつローカルエリアネットワーク (LAN) 内で行う．Kubernetes の機能を使用し，リバースプロキシから接続可能な Service を一時保存する．

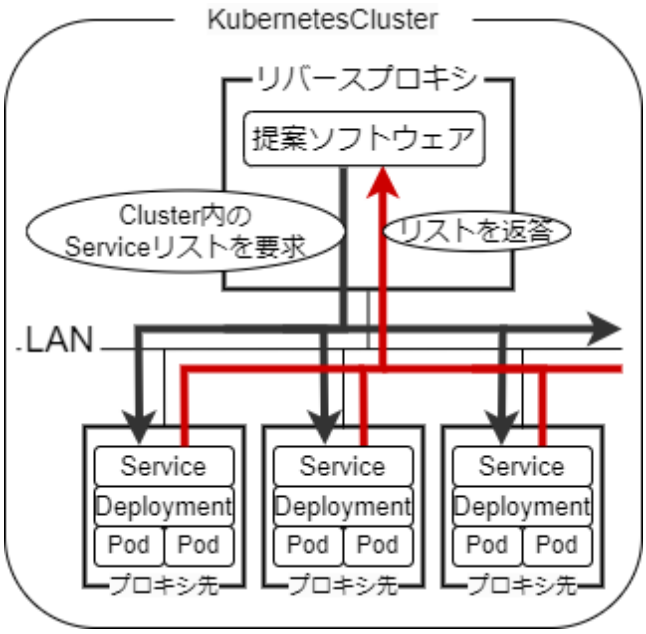

図 4 プロキシ先を探索

工程③では Nginx の設定情報を含めたマニフェストを生成し，リバースプロキシのコンテナを構築する．このマニフェストには，②で得た Service を全てプロキシ先とするように設定を書き込む．

工程④ではプロキシ状況の表示を行う．プロキシ先へのパス割当は自動的に行われ，リバースプロキシ構築完了後に割当てられたパスを表示する．

図 5 は提案ソフトウェアの流れを表したシーケンス図である．ライフラインはシステム管理者，リバースプロキシ用のサーバ，プロキシ対象のサーバ，これらの 3 つがありそれぞれのやり取りを示している．

主なライフライン同士のやり取りは以下である．システム管理者がリバースプロキシ用のサーバで提案ソフトウェアを実行すると，提案ソフトウェアは k3s をインストールする．次に，プロキシ先のアドレスとポート番号は Cluster 内の Service 一覧を問い合わせることで取得する．さらに，取得した Service 一覧を基にして nginx.conf を生成し，リバースプロキシを構築する．最後にリバースプロキシ用の

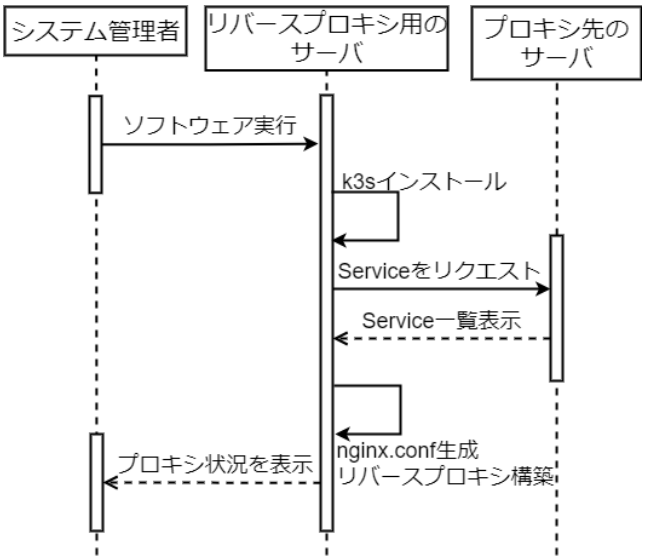

図 5 提案ソフトウェアの具体的な流れ

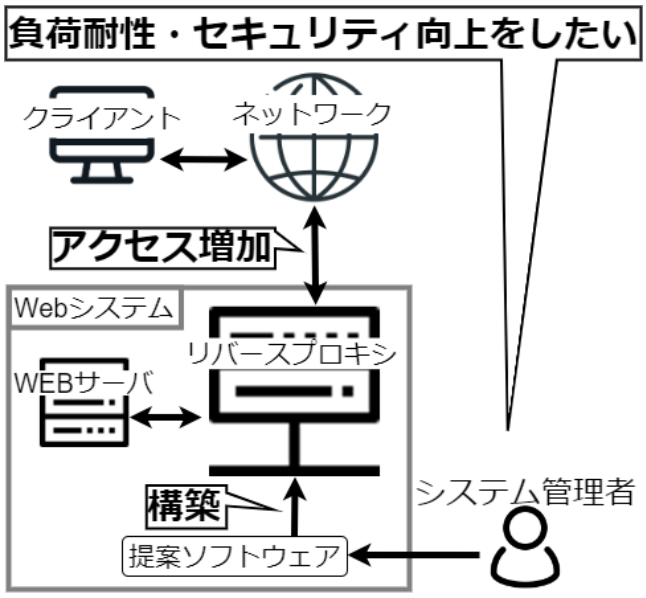

図 6 ユースケースシナリオ

サーバにプロキシの状況が表示され，システム管理者がこれを視認する．

ユースケース・シナリオ

本稿では，自社サーバ上で Web サイトを管理している食品製造会社をユースケースとする．ユースケースシナリオを図 6 に示す．システム管理者は自社サーバの設計・構築・運用・保守を行っている．自社の Web サーバ上では，主に会社概要や取り扱い商品を紹介するための Web ページを運用している．

ソーシャルネットワーキングサービス (SNS) が普及している現代では，ある日唐突に商品やサービスが SNS で話題に上がり，その商品やサービスに関係する Web ページのアクセスが増えることがある[*7]．アクセスの増加に伴い Web サーバへの負荷が増え，システム管理者が負荷耐性の向上を試みるケースがある．本稿ではそのような状況で，サーバの負荷耐性向上を目的したリバースプロキシ設置をすることを想定する．

システム管理者は必ずしも Nginx や Kubernetes に詳しいとは限らない．これらの知識に乏しい場合，Nginx を利用したリバースプロキシの構築には時間を要することが予想される．本稿で提案するソフトウェアは，プロキシ先のサーバとリバースプロキシのサーバを接続するまでの構築を行う．そのため Nginx の知識に乏しい初学者でも不明点の調査における時間を要さずリバースプロキシの構築ができる．

4. 実装

提案手法を基に，リバースプロキシの構築を自動で行うソフトウェアを新たに作成する．ソフトウェアを作成するにあたってプログラミング言語はシェルスクリプトと Python を使用する．実装の概要を図 7 に示す．

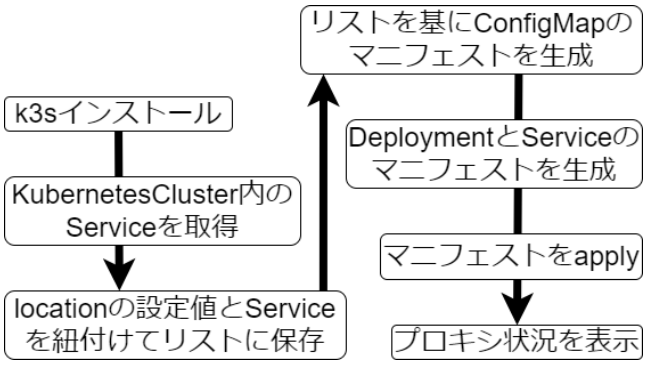

図 7 ソフトウェアの概要

まずコンテナ管理のために，k3s のインストールを行う．次に Kubernetes のコマンドを使用して，Cluster 内の存在する Service を取得する．取得した Service は，取得数に応じて番号を振り分けリストに保存する．

次に保存したリストを基に ConfigMap のマニフェストファイルを作成する．ソースコード 2 は ConfigMap の例である．ソースコード 2 の 1 行目から 6 行目までは，そのファイルが何の役割を持つマニフェストであるかが記述されており，例として 2 行目ではマニフェストの種類が ConfigMap であること，5 行目では構築されるアプリケーションのラベルが”nginx-app-proxy”であることが書かれている．7 行目の”data:”以降は，ConfigMap 特有のアプリケーション構成情報を配置する機能があるが，この構成情報を記述するための場所である．ソースコード 2 は，Nginx 設定ファイルの nginx.conf に書き込まれる情報が記述されている．本提案ソフトウェアでは，ソースコード 2 における 27 行目から 29 行目のようなプロキシ先の情報を

[*7] https://twitter.com/AJIMAI3/status/1600350358005764096

ソースコード 2　ConfigMap のサンプル

```
apiVersion: v1
kind: ConfigMap
metadata:
  labels:
    app: nginx-app-proxy
  name: nginx-default-conf
data:
  default.conf: |-
    error_log /var/log/nginx/error.log;
    server {
      listen 80;
      listen [::]:80;
      server_name localhost;
      location / {
        root /usr/share/nginx/html;
        index index.html index.htm;
      }
      error_page 500 502 503 504 /50x.html;
      location = /50x.html {
        root /usr/share/nginx/html;
      }
      proxy_set_header Host $host;
      proxy_set_header X-Real-IP $remote_addr;
      proxy_set_header X-Forwarded-Host $host;
      proxy_set_header X-Forwarded-Server $host
          ;
      proxy_set_header X-Forwarded-For
          $proxy_add_x_forwarded_for;
      location /test1 {
        proxy_pass http://192.168.100.166:31222/
            c0a20060.html;
      }
    }
```

記述する部分に，前過程でリスト保存した Service の情報をもとに動的に決定・記述し，マニフェストを生成する．

次の過程では Deployment と Service のマニフェストファイルを生成する．この際の生成は静的に行われる．

必要なマニフェストが全て生成されたため，次に全てのマニフェストを apply する．この apply でリバースプロキシの構築自体が完了する．

最後にプロキシ状況をユーザが視認出来るようにを表示する．この際，実際にアクセスするためのドメイン，各 Service，これらをシステム管理者が提案ソフトウェアを実行したターミナルに表示する．

5. 実験

本稿の課題としているリバースプロキシの構築作業が，初学者が行った場合に長時間を要するかを確かめる．今回は使用するサーバが，OS インストール直後の環境からリバースプロキシとして機能する状態になるまでの構築作業を手動で行い，所要時間を計測する．

実験環境

実験環境は，リバースプロキシ用の VM とプロキシ先の VM の 2 つを用いる．リバースプロキシの構築作業はリバースプロキシの構築未経験の人間が 1 人で行う．以下に VM の構成要素を示す．

- リバースプロキシを構築する VM 構成情報
  OS: Ubuntu-22.04
  vCPU: 1 コア
  RAM: 1GB
  HDD: 20GB
- プロキシ先の VM 構成情報
  OS: Ubuntu-22.04
  vCPU: 1 コア
  RAM: 1GB
  HDD: 20GB

実験結果と分析

図 8 はリバースプロキシを手作業で構築した作業時間を示したグラフである．構築には作業全体で 225 分かかった．内訳として k3s のインストールに 5 分，ConfigMap と Deployment のマニフェストファイルの記述にそれぞれ 10 分，Service マニフェストファイルの記述に 5 分，作成した 3 つのマニフェストファイルを apply するのに 5 分，構築したリバースプロキシが正常に稼働しているかの状態確認に 20 分，ConfigMap マニフェストファイルの誤りを試行錯誤しながら修正するのに 70 分，各工程の不明点を調べ理解するのに 100 分かかった．工程の中で ConfigMap マニフェストの修正が 31.11%，各工程の不明点の調査が 44.44%，合計で全体の 75.56%である．このことから構築作業が長時間になる最大の要因は，Nginx やサーバに関する知識不足である．

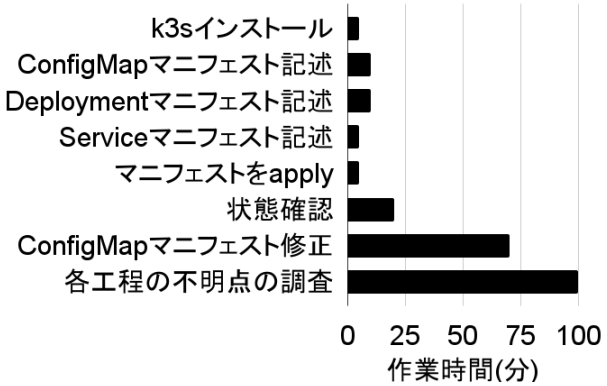

図 8　基礎実験の結果

6. 議論

本稿では，課題をリバースプロキシ構築に長時間を要することとし，作業の自動化を提案した．しかし本提案には，

非効率な処理を行う部分が存在する．何故なら本提案ソフトウェアは，実行開始から終了までの処理を連続して実行する．

例えば本提案ソフトウェアを使用して実際に OS インストール直後の VM にリバースプロキシを構築する場合，提案ソフトウェアを二度実行する必要がある．一度目の実行では，k3s インストール後に Service 探索を行ってもプロキシ先が存在しない．そのためソフトウェアを一度目実行後，プロキシ先となる Service を既存のサーバから移行または新規作成し，その後二度目のソフトウェア実行を行う必要がある．

これの解決策として，提案ソフトウェアの部分的処理の省略や再実行を可能とすることが有効である．具体的には，過去に k3s のインストールを行っていた場合のインストール処理の省略や，Service 探索部分からの途中開始をソフトウェア実行者が選択可能にすることが挙げられる．

7. おわりに

本稿の課題は，Nginx 初学者がリバースプロキシ構築に長時間を要することである．実際にリバースプロキシ構築未経験者が構築作業をした場合の基礎実験を行った．本稿では，リバースプロキシ構築過程である k3s インストール，プロキシ先確認，マニフェスト生成，マニフェストの apply，これらの自動化を提案とした．基礎実験の結果として，リバースプロキシ構築未経験者が手作業で構築したところ，完了までに 225 分掛かった．また，全体の 75.56% である 170 分は，各工程の不明点検索や誤ったマニフェストの修正に掛かった時間であり，これは知識不足が要因である．次回のレポートではこのデータを比較対象として評価を行う．

謝辞 本テクニカルレポートを執筆にあたりご指導いただきました東京工科大学コンピュータサイエンス学部先進情報専攻の三上智徳さん，圖齋雄治さん，増田和範さん，西嶌知良さんに御礼申し上げます．

参考文献

[1] Chyrvon, A., Lisovskyi, K. and Kyryndas, N.: THE MAIN METHODS OF LOAD BALANCING ON THE NGINX WEB SERVER, (online), DOI: 10.36074/logos-26.05.2023.040 (2023).

[2] Tedyyana, A. and Ghazali, O.: Teler Real-time HTTP Intrusion Detection at Website with Nginx Web Server, *JOIV : International Journal on Informatics Visualization*, Vol. 5, pp. 327–332 (online), DOI: 10.30630/joiv.5.3.510 (2021).

[3] Ma, C. and Chi, Y.: Evaluation Test and Improvement of Load Balancing Algorithms of Nginx, *IEEE Access*, Vol. 10, pp. 14311–14324 (online), DOI: 10.1109/ACCESS.2022.3146422 (2022).

[4] Tao, G. and Palaoag, T. D.: New Strategy of Communication System Server State Based on QSC Algorithm, *Revista Ibérica de Sistemas e Tecnologias de Informação*, No. E55, pp. 517–525 (2023).

[5] Pahl, C.: Containerization and the PaaS Cloud, *IEEE Cloud Computing*, Vol. 2, No. 3, pp. 24–31 (online), DOI: 10.1109/MCC.2015.51 (2015).

[6] Baur, A.: Packaging of kubernetes applications, *Universität Ulm*, (online), DOI: 10.18725/OPARU-38549 (2021).

[7] Rahman, A., Shamim, S. I., Bose, D. B. and Pandita, R.: Security Misconfigurations in Open Source Kubernetes Manifests: An Empirical Study, *ACM Trans. Softw. Eng. Methodol.*, Vol. 32, No. 4 (online), DOI: 10.1145/3579639 (2023).

[8] Kudo, R., Kitahara, H., Gajananan, K. and Watanabe, Y.: Application Integrity Protection on Kubernetes cluster based on Manifest Signature Verification, *Journal of Information Processing*, Vol. 30, pp. 626–635 (online), DOI: 10.2197/ipsjjip.30.626 (2022).

[9] Lee, S., Wong, T. and Kim, H. S.: To Automate or Not to Automate: On the Complexity of Network Configuration, *2008 IEEE International Conference on Communications*, pp. 5726–5731 (online), DOI: 10.1109/ICC.2008.1072 (2008).

[10] Wang, J. and Kai, Z.: Performance Analysis and Optimization of Nginx-based Web Server, *Journal of Physics: Conference Series*, Vol. 1955, No. 1, p. 012033 (online), DOI: 10.1088/1742-6596/1955/1/012033 (2021).

[11] Muntoni, G., Soldani, J. and Brogi, A.: Mining the Architecture of Microservice-Based Applications from their Kubernetes Deployment, *Advances in Service-Oriented and Cloud Computing* (Zirpins, C., Paraskakis, I., Andrikopoulos, V., Kratzke, N., Pahl, C., El Ioini, N., Andreou, A. S., Feuerlicht, G., Lamersdorf, W., Ortiz, G., Van den Heuvel, W.-J., Soldani, J., Villari, M., Casale, G. and Plebani, P., eds.), Cham, Springer International Publishing, pp. 103–115 (2021).