# Kubernetesを用いた講義における課題の進捗にもとづくPodの削除によるストレージ使用量の削減

鈴木 友也[1]　平尾 真斗[1]　串田 高幸[1]

**概要**：東京工科大学では，Kubernetes を用いた Site Reliability Engineering の講義を 3 年生に向けて行っている．講義では，3 人から 4 人で構成されたグループ内で仮想マシンを共有する．課題は，Pod が必要か不要か判断できないことである．判断できないと Pod を削除することができないため，講義の進行を止めてしまう．提案は，Pod の不必要を課題の進捗を管理しているスプレッドシートから判断し，不要な Pod を自動で削除することである．提案方式をもとに Python 3.10.12 を使用したソフトウェア，Google Apps Script を使用したソフトウェア，Google スプレッドシートを用いたスプレッドシートを作成した．評価実験は仮想マシンにユーザーを 4 人作成し，管理者である TA を 1 人と講義を受ける学生を 3 人とした．30 分で課題を行い，全学生がすべての課題を完了した．作成された全ての Pod，Deployment，Service，PVC，Secret が削除対象となった．課題終了後，ストレージは 9.8GB 使用されていた．作成されたものは，Pod が 72 個，Deployment が 6 個，Service が 6 個，PVC が 6 個，Secret が 3 個であった．ソフトウェア実行後，ストレージは 9.4GB 使用され，課題で作成された全ての Pod，Deployment，Service，PVC，Secret が削除されていた．ストレージ使用量は，約 4.1 %削減できた．全ての Pod，Deployment，Service，PVC，Secret が削除されたため，新しく Pod，Deployment，Service，PVC，Secret を作成する際に過去に作成された Pod，Deployment，Service，PVC，Secret が干渉することはなくなる．

## 1. はじめに

### 背景

東京工科大学では，Kubernetes[*1]を用いた Site Reliability Engineering[*2]（以下，SRE という）の講義を 3 年生に向けて行っている．1 回の講義は，約 5 時間あり，最初の約 30 分で SRE の説明を受け，残りの時間で課題を行う．課題の進捗は，スプレッドシートで管理され，Student Assistant（以下，SA という）または Teaching Assistant（以下，TA という）がチェックをする．グループ全員が課題が終了したら，SA または TA が確認し，スプレッドシートで対象者の完了した課題の未完了を完了に変更する．講義時間内に終わらなかった課題は，宿題となる．2023 年度の前期の講義は，40 人の学生が受講していた．第 1 回から第 7 回までは 3 人のグループが 4 個，4 人のグループが 7 個，第 8 回から第 14 回までは 4 人のグループが 10 個で構成されていた．講義が半分終了した第 8 回からグループを変えて講義を行った．

Kubernetes は，コンテナ化されたワークロードやサービスを管理するためのオープンソースのプラットフォームである [1–3]．Kubernetes をデプロイすると，クラスタが展開される．Kubernetes クラスターは，ノードの集合である．ノードには，マスターノードとワーカーノードがある．コントロールプレーンには，ワーカーノードと Pod を管理するための API が含まれている．Pod は 1 つ以上のコンテナのグループである．Pod は，Kubernetes で作成および管理できるデプロイ可能なコンピューティングの最小単位である [4]．

それぞれのグループに，仮想マシンが 2 つずつ与えられる．1 つは Kubernetes をインストールし，マスターノードとして使用する．この仮想マシンはターゲットサーバーになり，Group1 では sre001t という名前が付けられている．Group2 の場合は sre002t となり，グループ番号ごとに数字が変わる．もう 1 つは，学生が SSH し，Pod を作成するためのファイルを作成し，kubectl コマンドを動かすために使用する．この仮想マシンはホストサーバーになり，Group1 では sre001h という名前が付けられている．Group2 の場合は sre002h となり，グループ番号ごとに数字が変わる．初めて仮想マシンに SSH すると，SSH したユーザーのホームディレクトリが作成される．第 8 回からの新しいグループでは，第 7 回までのグループで使われて

[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
[*1] https://kubernetes.io/ja/
[*2] https://sre.google/

いた仮想マシンをそのまま使う．そのため，第 8 回からは第 7 回までに作成された Pod やファイルがそのまま残されている．

**課題**

課題は，Pod が必要か不要か判断できないことである．Pod が必要か不要か判断できないと，Pod を削除することができない．必要な Pod を削除してしまうと，過去に作成された Pod が原因で新しく Pod を作成できないことがあるため，Pod を削除できないことは講義の進行を止めてしまう．東京工科大学の講義では複数人が 1 つの仮想マシンで作業をする．図 1 は Group1 の仮想マシンを示す．

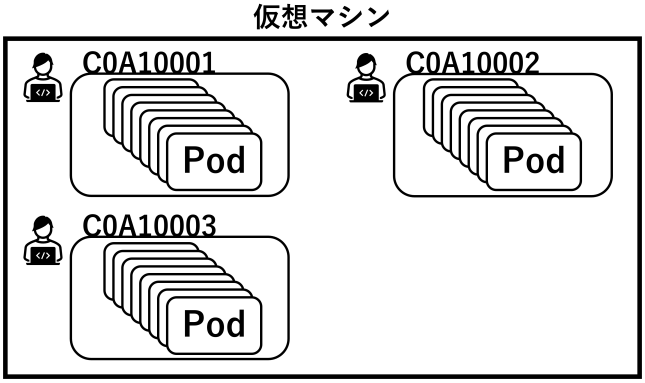


**図 1** Group1 の仮想マシン

グループが変わる第 8 回では，第 7 回までのグループで使われていた仮想マシンの Pod がそのまま残されていて，図 2 のように 1 つの仮想マシンに多くの Pod が存在してしまう．Pod の数が増えると，Pod を作成できない問題が起こりやすくなる．

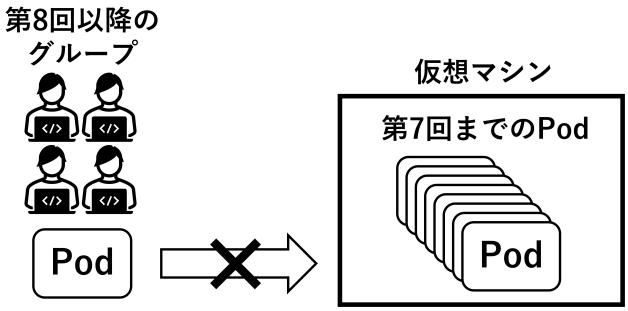


**図 2** 第 8 回に起こる問題

**各章の概要**

第 2 章の関連研究では，関連する既存研究を述べる．第 3 章の提案では，課題を解決する提案方式，ユースケース・シナリオの説明をする．第 4 章の実装では，実装方法の説明する．第 5 章の評価実験では，実験環境，実験結果と分析の説明をする．第 6 章の議論では，提案方式の議論をする．第 7 章の結論では，全体をまとめる．

## 2. 関連研究

Kubernetes クラスタにおける Pod のデプロイの最適解ツールである SAGE を提案している研究がある [5]．SAGE はデプロイされるアプリケーションの制約や使用可能なクラウドリソースにもとづいて最適なデプロイ計画を計算し Pod の配置を行う．一方で，Kubernetes のクラスタ内に配置されている未使用の Pod を削除する機能がない．そのため Pod が追加されていくとクラスタ内のリソースが枯渇し，結果的に Pod が立てられなくなる．

QoS 保証を前提としてリソースの無駄を削減できる，Kubernetes クラスタの規模を動的に調整する汎用システムを提案している研究がある [6]．このシステムでは，監視モジュール，QoS モジュール，スケーリングモジュール，および実行モジュールの 4 つのモジュールがある．この 4 つのモジュールを使うことで Kubernetes クラスタの規模の調節を行う．この提案システムは，Kubernetes クラスタを数に制限がなくクラスタを増やしても良い場合には授業でも適応できる．一方で授業で扱うクラスタの数には限りがあるため適応できない．授業では，Kubectl コマンドを打つための h サーバ，Kubernetes クラスタの Master である t サーバ，ワーカノードである s サーバしか使用できない．

クラウドコンピューティングにおける効率的な自動リソース管理のために，Petri Net ベースのパフォーマンスモデルを通じて実現される，Kubernetes 上のパフォーマンス分析を行った研究がある [7]．この研究では Pod あたりのコンテナ数の影響を理解することができる。それによりキャパシティプランニングとリソース管理，アプリケーションの設計，特に Pod とコンテナの観点からアプリケーションをどのように構築するかを考えるための基礎として活用できる．一方で，複数のコンテナが同じ計算リソースを奪い合う際に現れるコンテナ内でのリソース競合現象の分析ができない．複数人で 1 つの仮想マシンを扱う場合，リソースの競合が起こる場合がある．

## 3. 提案

**提案方式**

本提案方式では，必要な Pod と不要な Pod を課題の進捗にもとづいて区別し，不要な Pod を削除する．必要な Pod は，課題が完了していなく宿題として残されている Pod，次回以降も使うため残されている Pod である．不要な Pod は，課題が完了しているにもかかわらず削除されていない Pod である．課題の進捗を管理しているスプレッドシートを参照し，課題の完了，未完了を判断する．次回以降も使う Pod は，個人で作業する Namespace ではなく，そのための共有の Namespace で作業する．そのため，Namespace から判断することができる．図 3 に提案

ソフトウェアの概要を示す．提案ソフトウェアでは，スプレッドシートを読み取り，不要な Pod を判別する．最後に不要な Pod をコマンド履歴を参照して削除する．

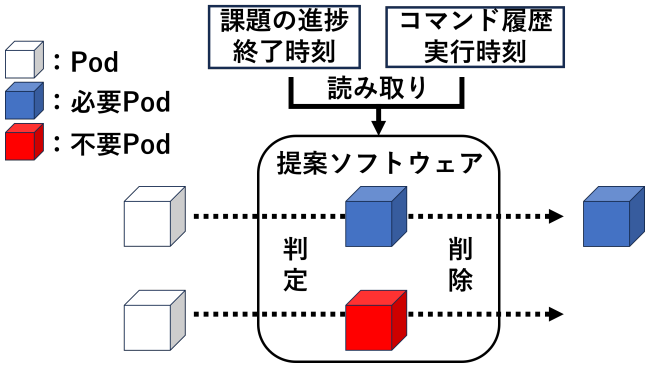


**図 3**　提案ソフトウェアの概要

**必要な Pod** は，課題が完了してしていないため宿題として残されている Pod と次回以降も使うため残されている Pod である．課題が完了していないかは課題の進捗を管理しているスプレッドシートから判断する．次回以降も使う Pod は Namespace から判断する．次回以降も使う Pod は個人で作業する Namespace ではなく，そのための共有の Namespace で作業する．実際に，2023 年度前期では，ログ検索に用いられるソフトウェアである Elasticsearch を使用した [8–10]．Elasticsearch は，第 11 回から第 14 回にかけて使用した．また，2023 年度後期では，Prometheus を使用した [11]．Prometheusu は第 4 回から第 5 回にかけて使用した．そのため，Elasticsearch と Prometheus は次回以降も使う Pod に分類される．これらは，ユーザーの学籍番号で作成された個人の Namespace ではなく，そのために作成された Namespace に作成した．

**不要な Pod** は，課題が完了しているにもかかわらず，削除されていない Pod である．課題が完了した場合，以降使うことがないため削除対象となる．課題が完了しているかは，課題の進捗を管理しているスプレッドシートから判断する．

**読み取り**

読み取りでは，課題の進捗，課題の終了時刻，コマンド履歴，コマンドの実行時刻を読み取る．東京工科大学の講義では，課題の進捗をスプレッドシートで管理しているが，スプレッドシートと Pod に関連性がないため，どの Pod がその課題で使われたものなのか判断できない．そこで，スプレッドシートに編集が行われたときに，その時刻と内容をスプレッドシートに記録する．また，コマンドの実行時刻も仮想マシン内に記録する．時刻を比較することによって課題で使われた Pod を特定する．

**判定**

判定では，課題の進捗をもとにして Pod の不必要を判断する．読み取りで，取得した課題の進捗を参照し，完了している課題の Pod を課題の終了時刻とコマンドの実行時刻から特定する．

**削除**

削除では，スプレッドシートとコマンドの実行履歴をもとにして不要と判断された Pod を削除する．作業途中の Pod は削除せずに残しておく．

**ユースケース・シナリオ**

図 4 にユースケースの概要を示す．東京工科大学で行われている，Kubernetes を用いた SRE の講義をユースケースとする．TA は授業全体の管理を行なっている．学生は Kubernetes を用いた授業を受けている．学生は仮想マシン上に作成された Kubernetes クラスター内に自分の学籍番号と同じ Namespace を作成する．それぞれ C0A10001，C0A10002，C0A10003 とする．各 Namespace 内には授業内で学生が授業で作成した Pod が配置される．使用 Pod は，授業内で課題が完了していない場合に残されている Pod である．未使用 Pod は，授業内で課題が完了している場合に残っている Pod である．Kubernetes を用いた授業では，未使用の Pod が消し忘れで配置されている場合があり，次の課題を進める際に Pod が立たない原因となる．その際に提案のソフトウェアを用いれば，授業内で使用しない Pod を削除することができる．その結果 Pod が作成できない原因を回避できる．

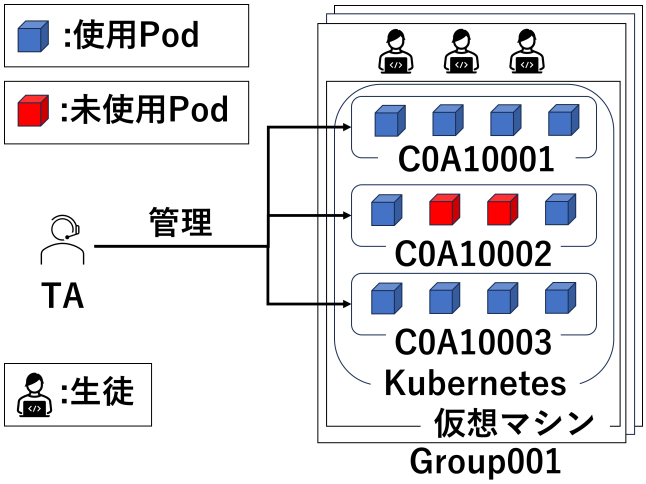


**図 4**　ユースケースの概要

## 4.　実装

提案方式をもとに Python 3.10.12 を使用したソフトウェア，Google Apps Script を使用したソフトウェア，Google スプレッドシートを用いたスプレッドシートを作成した．Python 3.10.12 を使用したソフトウェアには，仮想マシンの各ユーザーのコマンド履歴の取得，スプレッドシート

の読み取り，Pod の削除を行う機能がある．Google Apps Script を使用したソフトウェアには，スプレッドシートの編集履歴をスプレッドシートに記録する機能がある．Google スプレッドシートを用いたスプレッドシートには，課題の進捗とスプレッドシートの編集履歴が記述されている．

図 5 に読み取りの概要を示す．仮想マシン内の各ユーザーが持つ，.bash_history からコマンドの履歴とコマンドの実行時刻を読み取り，外部にあるスプレッドシートから課題の進捗と課題の完了時刻を読み取る．スプレッドシートから読み取ったものは，JSON ファイルとして保存される．

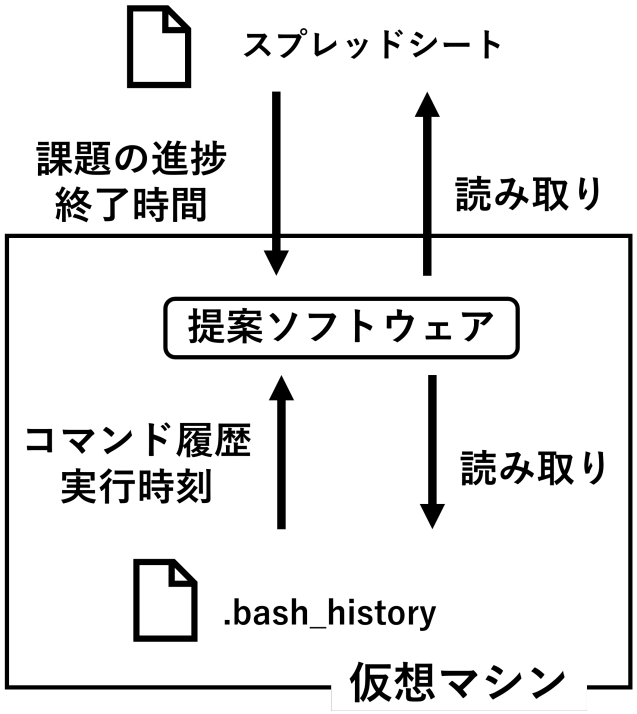


**図 5**　読み取りの概要

コマンド履歴は Linux のユーザーが持つ，.bash_history から取得する．.bash_history には，コマンドの履歴が記録されている．本提案方式では，コマンドの実行時刻も利用するため，.bash_history に時刻も記録されるようにする必要がある．コード 1 を記述した/etc/profile.d/history.sh を作成する．仮想マシンを利用しているユーザー全員の.bash_history に時刻が記録されるようになる．

スプレッドシートの読み取りは，Google Sheets API を使用する．https://sheets.googleapis.com/v4/spreadsheets/スプレッドシート ID/values/シート名?key=API キーにアクセスすると，JSON のデータが表示される．仮想マシンから curl コマンドでアクセスし，JSON ファイルに書き込む．

**コード 1**　/etc/profile.d/history.sh

```
1 HISTTIMEFORMAT='%F %T ';
```

図 6 に Pod の削除方法を示す．Pod の削除は，課題が終了した時刻までに実行されたコマンドを参照し，kubectl apply コマンドを kubectl delete コマンドに変更して実行する．課題終了時刻を過ぎて実行されたコマンドは，まだ完了していない課題で実行されたコマンドであるため，そのコマンドで作成された Pod は削除しない．

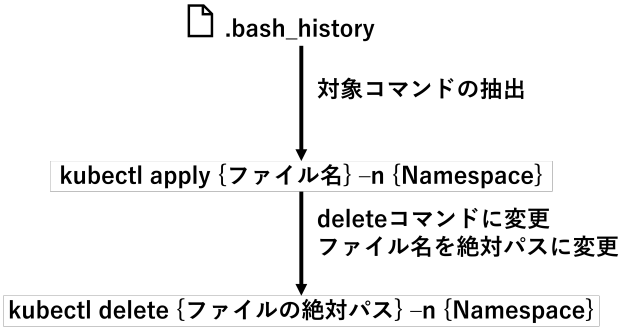


**図 6**　Pod の削除方法

図 7 に進捗管理用スプレッドシートの概要を示す．1 番左の列にグループ番号，左から 2 番目の列に学籍番号，左から 3 番目以降の列に課題の番号と進捗が示されている．課題が完了していることがチェックされたら，未完了を完了に変更する．

| Group | 学籍番号 | 1-1 | 1-2 |
|---|---|---|---|
| Group1 | C0A10001 | 完了 | 未完了 |
| | C0A10002 | 完了 | 未完了 |
| | C0A10003 | 完了 | 未完了 |
| Group2 | C0A10004 | 未完了 | 未完了 |
| | C0A10005 | 未完了 | 未完了 |
| | C0A10006 | 未完了 | 未完了 |
| | C0A10007 | 未完了 | 未完了 |

**図 7**　進捗管理用スプレッドシートの概要

図 8 に変更履歴のスプレッドシートの概要を示す．進捗管理用スプレッドシートが編集されると一番上の行に編集内容が記述される．左から，完了時刻，学籍番号，課題番号，編集後のセルの内容が記述される．

| | A | B | C | D |
|---|---|---|---|---|
| | 2023-12-20 09:48:57 | C0A10003 | 1-2 | 完了 |
| | 2023-12-20 09:48:56 | C0A10002 | 1-2 | 完了 |
| | 2023-12-20 09:48:53 | C0A10001 | 1-2 | 完了 |
| | 2023-12-20 09:44:24 | C0A10003 | 1-1 | 完了 |
| | 2023-12-20 09:44:20 | C0A10002 | 1-1 | 完了 |
| | 2023-12-20 09:44:19 | C0A10001 | 1-1 | 完了 |

**図 8**　変更履歴のスプレッドシートの概要

## 5. 評価実験

### 実験環境

c0a21069-sre001t と c0a21069-sre001h という名前の Ubuntu 22.04 が搭載された仮想マシンを使用した．表 1 に仮想マシンの User と役割を示す．仮想マシンに User を 4 人作成し，管理者である TA を 1 人と講義を受ける学生を 3 人とした．

**表 1** 仮想マシンの User と役割

| User | 役割 |
|---|---|
| c0a21069 | TA |
| c0a10001 | 学生 |
| c0a10002 | 学生 |
| c0a10003 | 学生 |

学生 3 人に，実際の講義を参考にした課題 1-1 と課題 1-2 を行ってもらい，複数の Pod，Deployment，Service，PVC，Secret を作成してもらった．表 2 に課題 1-1 で作成したファイルを示す．表 3 に課題 1-2 で作成したファイルを示す．

**表 2** 課題 1-1 で作成したファイル

| ファイル名 |
|---|
| mysql-pvc.yaml |
| mysql-secret.yaml |
| mysql-deployment.yaml |
| mysql-service.yaml |

**表 3** 課題 1-2 で作成したファイル

| ファイル名 |
|---|
| wordpress-pvc.yaml |
| wordpress-deployment.yaml |
| wordpress-service.yaml |

課題は 30 分間行った．図 9 に課題の結果を示す．全学生が 30 分で課題が完了したため，作成された全ての Pod，Deployment，Service，PVC，Secret が削除対象となった．


**図 9** 課題の結果

### 実験結果と分析

図 10 にストレージ使用量の比較を示す．ストレージは，/dev/mapper/ubuntu–vg-ubuntu–lv を参照した．課題終了後，ストレージは 9.8GB 使用されていた．作成されたものは，Pod が 72 個，Deployment が 6 個，Service が 6 個，PVC が 6 個，Secret が 3 個であった．ソフトウェア実行後，ストレージは 9.4GB 使用され，課題で作成された全ての Pod，Deployment，Service，PVC，Secret が削除されていた．ストレージ使用量は約 0.4GB 削減された．結果として約 4.1 %削減できた．全ての Pod，Deployment，Service，PVC，Secret が削除されたため，新しく Pod，Deployment，Service，PVC，Secret を作成する際に過去に作成された Pod，Deployment，Service，PVC，Secret が干渉することはなくなる．

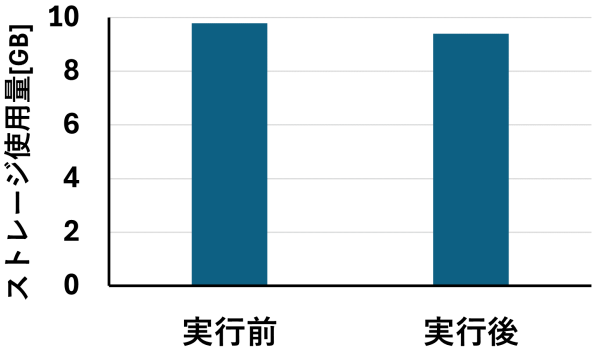


**図 10** ストレージ使用量の比較

## 6. 議論

講義内で Prometheus や ElasticSearch の Pod は共有の Namespace 内に作成した．本稿ではこの Namespace は複数回の講義を跨いで使うため全て必要 Pod とした．しかし，講義が進行し，これらの Pod が残っている場合，Pod が作成できない原因につながる．そのためこれらの Pod を講義内で使わない場合，共有の Namespace 内の Pod も削除する必要がある．共有の Pod を消す方法として PDF 化されている講義資料を解析し，Pod の不必要を判断する方法がある．課題が切り替わり，今まで使っていたものを使わなくなったことを確認したらその Pod を不要と判断する．PDF は Python を用いて読み取ることができる[*3]．

講義の構成によっては，課題が完了していても削除してはいけないものがある．自分が完了していても，課題が完了していない他のグループメンバーが使用するために作成したものを削除できない場合がある．スプレッドシートの項目を完了，未完了の他に削除不可を作り，削除対象の決定方法を変更することで解決できる．

本稿では最後に完了した課題以前に実行された kubectl

[*3] https://qiita.com/mima_ita/items/d99afc28b6f51479f850

apply を kubectl delete に変更して Pod の削除を行っている．そのため，前回までに未完了の課題が残っている場合その Pod を削除してしまう．講義時間外で実行されたコマンドを対象外にすることで，未完了の Pod を残すことができる．また，既に削除されている Pod に対しても kubectl delete コマンドを実行している．最後に kubectl apply を実行している Pod を削除対象にし，最後に kubectl delete を実行している Pod は削除対象にしないことで，不要な delete コマンドを実行しなくなる．

評価実験では，作成した Pod をわざと残し，削除した．実際の講義では，学生が削除し忘れた際に，不要な Pod が残る．実際の講義で運用することで，自然な環境で実験ができる．

## 7. 結論

東京工科大学では，Kubernetes を用いた Site Reliability Engineering の講義を 3 年生に向けて行っている．講義では，3 人から 4 人で構成されたグループ内で仮想マシンを共有する．課題は，Pod が必要か不要か判断できないことである．判断できないと Pod を削除することができないため，講義の進行を止めてしまう．提案は，Pod の不必要を課題の進捗を管理している判断しスプレッドシートから判断し，不要な Pod を自動で削除することである．提案方式をもとに Python 3.10.12 を使用したソフトウェア，Google Apps Script を使用したソフトウェア，Google スプレッドシートを用いたスプレッドシートを作成した．c0a21069-sre001t と c0a21069-sre001h という名前の Ubuntu 22.04 が搭載された仮想マシンを使用して評価実験を行った．仮想マシンにユーザーを 4 人作成し，管理者である TA を 1 人と講義を受ける学生を 3 人とした．30 分で課題を行い，全学生がすべての課題を完了した．作成された全ての Pod，Deployment，Service，PVC，Secret が削除対象となった．課題終了後，ストレージは 9.8GB 使用されていた．作成されたものは，Pod が 72 個，Deployment が 6 個，Service が 6 個，PVC が 6 個，Secret が 3 個であった．ソフトウェア実行後，ストレージは 9.4GB 使用され，課題で作成された全ての Pod，Deployment，Service，PVC，Secret が削除されていた．ストレージ使用量は，約 4.1 %削減できた．全ての Pod，Deployment，Service，PVC，Secret が削除されたため，新しく Pod，Deployment，Service，PVC，Secret を作成する際に過去に作成された Pod，Deployment，Service，PVC，Secret が干渉することはなくなる．

## 参考文献

[1] Balla, D., Simon, C. and Maliosz, M.: Adaptive scaling of Kubernetes pods, *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*, IEEE, pp. 1–5 (2020).

[2] Carrión, C.: Kubernetes scheduling: Taxonomy, ongoing issues and challenges, *ACM Computing Surveys*, Vol. 55, No. 7, pp. 1–37 (2022).

[3] Luksa, M.: *Kubernetes in action*, Simon and Schuster (2017).

[4] Zhu, M., Kang, R., He, F. and Oki, E.: Implementation of Backup Resource Management Controller for Reliable Function Allocation in Kubernetes, *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*, pp. 360–362 (online), DOI: 10.1109/NetSoft51509.2021.9492724 (2021).

[5] Luca, V.-I. and Erascu, M.: SAGE–A Tool for Optimal Deployments in Kubernetes Clusters, *arXiv preprint arXiv:2307.06318* (2023).

[6] Wu, Q., Yu, J., Lu, L., Qian, S. and Xue, G.: Dynamically Adjusting Scale of a Kubernetes Cluster under QoS Guarantee, *2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 193–200 (online), DOI: 10.1109/ICPADS47876.2019.00037 (2019).

[7] Medel, V., Tolosana-Calasanz, R., Bañares, J. Á., Arronategui, U. and Rana, O. F.: Characterising resource management performance in Kubernetes, *Computers & Electrical Engineering*, Vol. 68, pp. 286–297 (2018).

[8] Kathare, N., Reddy, O. V. and Prabhu, V.: A comprehensive study of elasticsearch, *International Journal of Science and Research (IJSR)* (2020).

[9] L'Hôte, A. and Jeangirard, E.: Using Elasticsearch for entity recognition in affiliation disambiguation, *arXiv preprint arXiv:2110.01958* (2021).

[10] Zamfir, V.-A., Carabas, M., Carabas, C. and Tapus, N.: Systems Monitoring and Big Data Analysis Using the Elasticsearch System, *2019 22nd International Conference on Control Systems and Computer Science (CSCS)*, pp. 188–193 (online), DOI: 10.1109/CSCS.2019.00039 (2019).

[11] Chen, L., Xian, M. and Liu, J.: Monitoring System of OpenStack Cloud Platform Based on Prometheus, *2020 International Conference on Computer Vision, Image and Deep Learning (CVIDL)*, pp. 206–209 (online), DOI: 10.1109/CVIDL51233.2020.0-100 (2020).