KubernetesにおけるServiceの作成順をもとにしたPort番号の変更による重複の解消

近藤 悠斗[1]　平尾 真斗[2]　串田 高幸[1]

概要：東京工科大学では 3 年生に向けて Site Reliability Engineering の実習を行っている．実習では，4 人で構成されたグループごとに 3 台の仮想マシンを使用する．学生は Kubernetes を操作し Service を作成して実習課題を進める．授業時間内に実習課題が完了しなかった学生はその内容を宿題として次回の授業開始時までに取り組む．課題は宿題が完了せず作成されたままになっている Service と実習課題を完了するために作成する Service で Port 番号が重複し Service の作成ができないことである．本稿では，Service に重複が発生したときに先に作成されていた Service の名前と Port 番号を保存し，Port 番号を変更することでその Port で新しく Service を作成できるようにし，Service 削除するコマンドが送信された際にファイルから以前にその Port を使用していた Service を特定し，それを復元する方法を提案した．提案方式をもとに Service 名と Port 番号の保存と，Port 番号の変更を行うソフトウェアを構築する．基礎実験では実際に第 4 回から第 12 回の実習で，実習開始時に全グループの Service を取得した．実習では各グループでそれぞれ 7 個 Port 番号が使用できる．そのため，4 人グループの場合，Service が 4 つ以上残っているとさらに実習課題の達成のため 1 人 1 つの Service を作成すると Port 番号が足りなくなってしまう．第 4 回から 12 回の全グループで，残っている Service が 4 つ以上ものは 90 件中 38 件だった．

1. はじめに

背景

東京工科大学では 3 年生に向けて Site Reliability Engineering の実習を行っている [1, 2]．その実習では Kubernetes を使用し Web アプリケーション，ロギングソフトウェアを構築する実習を行っている [3]．例えば WordPress や Nginx である [4, 5]．実習は全 14 回で構成されており，毎週 1 回，各 5 時間行っている．2024 年度の前期の実習は，40 人の学生が受講しており，各グループ 4 人ずつの計 10 グループで実習課題に取り組んだ．実習課題の進捗はスプレッドシートで管理されている．Student Assistant(以下 SA) が各チームで実習課題の完了を確認し，Teaching Assistant(以下 TA) が SA から報告を受けてスプレッドシートに進捗を記入する．

Kubernetes は現在最も使用されているコンテナオーケストレーションフレームワークである [6–9]．Kubernetes をサーバに展開するとクラスタが作成される．クラスタはノードの集合で，マスターノードとワーカーノードが含まれる．Service はクラスター内で 1 つ以上の Pod として実行されているネットワークアプリケーションを公開するための Kubernetes リソースである [10, 11]．

実習の各グループには仮想マシン (以下 VM) が 3 つずつ与えられる [12, 13]．3 つの VM はグループ 1 なら sre001h(以下 h サーバ)，sre001t(以下 t サーバ)，sre001s(以下 s サーバ) と名付けられている．001 の部分にはグループ 2 ならば 002 のようにグループ番号が入る．h サーバは Kubectl コマンドを実行するサーバとして使用し学生はこの VM に SSH し自身の学籍番号のユーザでログインする [14, 15]．学生は授業開始時間から約 30 分間 TA から，実習資料の説明を聞く．その後，実習課題の完了のため YAML 形式のファイルを作成し，自身のネームスペースで Pod と Service の作成を行う．t サーバはマスターノードとして使用し，s サーバはワーカーノードとして使用する．

各授業回内で実習課題を達成できなかった学生は，宿題として翌週の授業までに実習課題を完了する必要がある．宿題を完了した学生はメールで実習課題で求められた Pod と Service を作成し，出力結果をスクリーンショットで撮影する．その後，画像を PDF に貼り付け，それを Moodle のアップロード先に提出する [16, 17]．

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

課題

課題は，宿題が完了せず残している Service と実習課題の完了のために作成した Service の Port 番号が衝突することによって Service が作成できないことである．実習の流れを図 1 に示す．

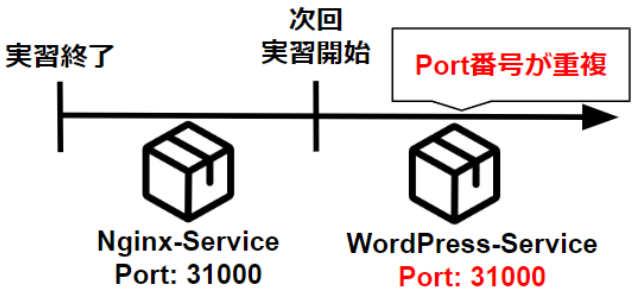

図 1: 実習の流れ

Nginx-Service は宿題を完了するために作成した Nginx の Pod にアクセスできるようにするための Service である．WordPress-Service は実習課題を完了するために作成した WordPress の Pod にアクセスできるようにするための Service である．

実習中に実習課題が完了しなかった学生は宿題として実習時間外に実習課題に取り組む．宿題がある学生は，実習終了後に自身のグループの h サーバにアクセスし，宿題を行う．t サーバには宿題を完了するための Pod と Service が作成される．図では実習終了後に Port 番号 31000 番を使用して Service を作成している．学生は次回の実習時までに宿題が完了しなかった Pod と Service を削除しない．そのため実習中に実習課題を完了するために作成した Service と Port 番号が重複し，Service が作成されなくなる．図では実習の Service を Port 番号 31000 番で作成しようとしており Port 番号が重複したため作成できていない．

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，実装方法の説明をする．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 章のおわりにでは，全体をまとめる．

2. 関連研究

大学のコンピュータサイエンス研究室に向けた Kubernetes のスケジューリング戦略を提案した研究がある [18]．この研究ではディープラーニングを行う Job を対象としており，GPU を多く使用するタスクに対して，タスクの完了速度と GPU の占有時間をもとにしてどの Pod を優先的にスケジューリングするかを決めている．そのためこの研究はバッチ処理を行うための一次的に作成され処理が終了した後は自動的に削除される Pod を対象にしており，Service の作成と削除については対象としていない．

クラスタの規模を動的に調整するシステムを提案している研究がある [19]．この研究では QoS を保証するために CPU 使用率やレイテンシを取得し，ノードの数を調整する．そのためこの研究はノードの数に制限が無い場合に適用できる．対して実習では各チームに 3 つの VM が割り当てられており，ノードは 2 つまでしか使用できないため適用できない．

Kubernetes の負荷分散とためにリーダをノード全体に分配する方法を提案している研究がある [11]．この論文では 1 種類のレプリカを公開するための LoadBalancer の分配について言及している，一方で Service 同士の優先順位についても重要である．

モバイルネットワークの通信の複雑性に対してネットワークサービスメッシュを使用することを提案した研究がある [20]．この研究ではレイヤー 3，レイヤー 4 レベルでの NAT について言及している．そのため Port 番号については対象としていない．

3. 提案

提案方式

本稿では，Service の作成された順番と Port 番号をテキストファイルに保存し，先に作成された Service の Port 番号を変更することで，重複した Port 番号を使用可能にする方法を提案する．

提案の概要を図 2 に示す．提案ソフトウェアは，本稿で提案する Service が配置されている Namespace，Service 名，Port 番号，その Service がその Port で何番目に作成されたかの番号，Service の種類の保存と，Port の変更を行うソフトウェアである．Nginx-service は，Nginx の Pod アクセスできるようにするために 31000 番の Port 番号を指定して作成された Service である．WordPress-service は，WordPress の Pod アクセスできるようにするために 31000 番の Port 番号を指定して作成しようとしている Service である．人のマークがついている Service は学生のコマンド入力によって作成されたものを指す．本提案は Service が作成されたタイミング，Service の Port 番号が重複したタイミング，Service が削除されたタイミングで動作する．図 2 でははじめに Nginx-Service が作成されたタイミングで Service が配置されている Namespace，Service 名，Port 番号，その Service がその Port で何番目に作成されたかの番号，Service の種類がテキストファイルに保存される．

次に実習中に WordPress-service を学生が作成しようとし，Port 番号が Nginx-service と重複しているため，作成に失敗している．この時，提案ソフトウェアは Nginx-service の Port 番号を別の番号に変更することによって，使用さ

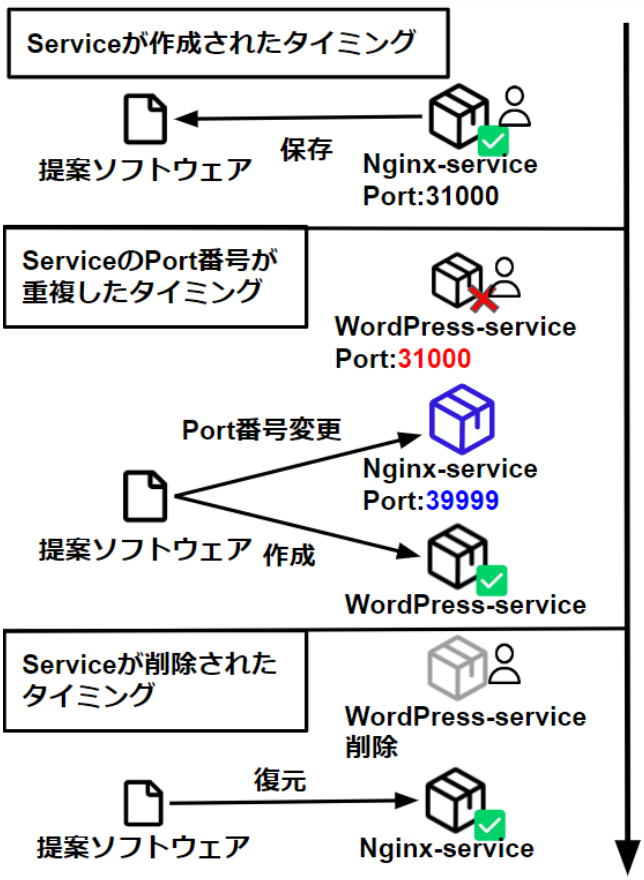

図 2: 提案の概要

れていた Port 番号を使えるようにする．その後，学生が Port 重複時に作成しようとしていた WordPress-service を起動するコマンドを送信し作成する．

最後に学生が WordPress-service を削除したタイミングで，提案ソフトウェアは Nginx-service を復元し，もとの Port 番号で動作するようにする．学生が Service を削除したタイミングは kubectl delete というコマンドを送信したタイミングとする．

Service が作成されたタイミング

まず Service が作成されたタイミングの処理について図 3 に示す．

図 3 では C0A10001 という Namespace，Nginx-service というサービス名，31000 という Port 番号，1 という作成順番号，NodePort というサービスの種類を保存している．Nginx-service は学生が作成した Nginx の Pod にアクセスするための NodePort の Service で Port 番号は 31000 番に設定されている．提案ソフトウェアは，本稿で提案する Service の情報の保存と Port の変更を行うソフトウェアである．サービス情報ファイルは作成されたタイミングで Service が配置されている Namespace，Service 名，Port 番号，その Service がその Port で何番目に作成されたかの番号，Service の種類を保存するためのテキストファイルで

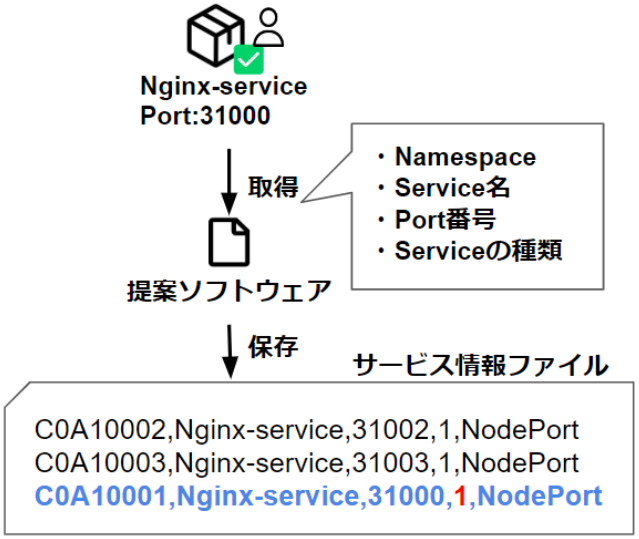

図 3: Service が作成されたタイミング

ある．

Service が作成されたタイミングで Namespace，Service 名，Port 番号，サービスの種類を取得する．これを Service の情報とする．次に提案ソフトウェアは Service の情報をファイルに保存する．その際，現在ファイルに保存されている Port 番号を参照し，保存されていなければこの Service が初めてその Port 番号を使用するとして 1 という番号を付与する．これを作成順番号とする．サービス情報ファイルには Namespace，Service 名，Port 番号，作成順番号，Service の種類が保存される．例えば図 3 では C0A10001 という Namespace，Nginx-service というサービス名，31000 という Port 番号，1 という作成順番号，NodePort というサービスの種類が保存されたことになる．

Service の Port 番号が重複したタイミング

次に Service の Port 番号が重複したタイミングの処理について図 4 に示す．

Service を作成しようとしたときに Port 重複のエラーがコマンドラインに出力された場合，その Service が使用しようとしている Port 番号を取得する．使用しようとしている Port 番号が保存したサービス情報ファイルに含まれていた場合，重複して作成に失敗した Service 自体を除いた作成順番号が最大の物を探す．図 4 では，まず WordPress-service を作成しようとしたが 31000 番の Port 番号がすでに使用されており，Service の作成に失敗している．その後，保存したサービス情報ファイルから 31000 番を使用している Service を特定する．この場合 Nginx-service が 31000 番の Port を使用しており，WordPress-Service は作成に失敗した Service のため，WordPress-Service を除くと Nginx-service 作成順番号が 1 で最大のため，この Service が現在この Port を使用していると分かる．

その後 Service の Port 番号を現在使用されていない番号

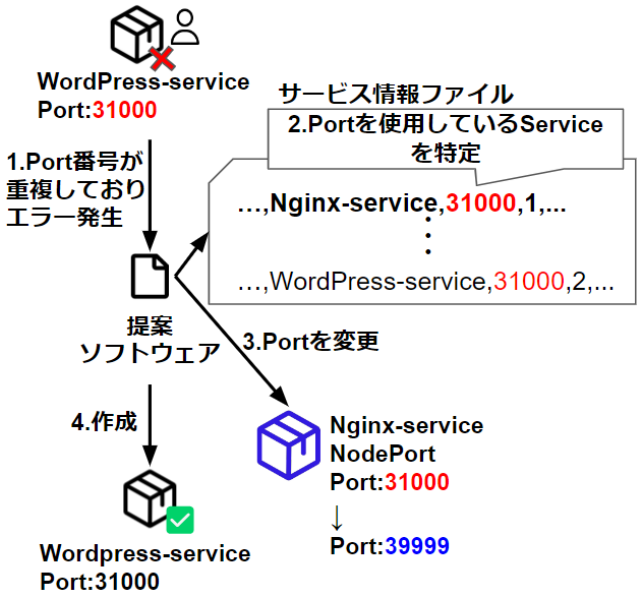

図 4: Service の Port 番号が重複したタイミング

に変更することによって，Port 番号の重複を解消する．この際ユーザにコマンドラインでサービス名と変更後の Port 番号を出力することで，Port 番号が変更されたことを通知する．ユーザ通知する出力内容をプログラム 1 に示す．

プログラム 1: ユーザ通知する出力内容

```
1 > Nginx-ServiceのPort番号は39999番に
  変更されました
```

Nginx-Service は Port を変更した Service 名を示す．39999 は変更後の Port 番号を示す．最後に作成できなかった Service を作成するコマンドを history から取得して再度実行し，Service を作成する．作成後は Service が作成されたタイミングと同様の手順で Service が配置されている Namespace，Service 名，Port 番号，その Service がその Port で何番目に作成されたかの番号，Service の種類をサービス情報ファイルに保存する．図 4 では Wordpress-service は，Nginx-service の次に 31000 番を使用して Service が作成されたため，作成順番号は 2 になる．

Service が削除されたタイミング

次に Service が削除されたタイミングの処理について図 5 に示す．

コマンド入力を取得するシェルスクリプトを使用し，学生が kubectl delete から始まるコマンドを入力するのを検知する．これを Service が削除されたタイミングとする．Service が削除されたら，ファイルからその Service 名が書かれている行を探す．その行があれば，その行の内容を削除する．図 5 では学生が WordPress-service を

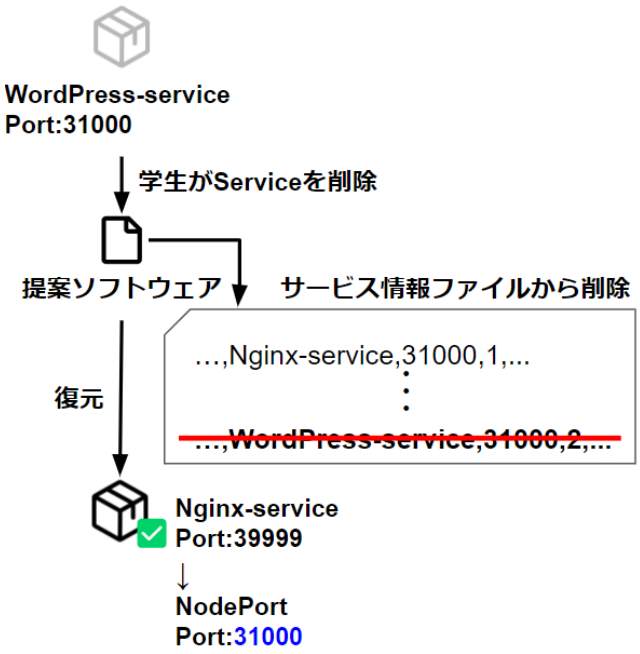

図 5: Service が削除されたタイミング

削除した状況である．この時提案ソフトウェアはファイルから WordPress-service を探し，その行を削除する．その後削除した行の作成順番号から 1 少ない物を探し，元の Port 番号に戻すことで復元を行う．図 5 では削除した WordPress-service の作成順番号が 2 のため，Port31000 番を使用していて，作成順番号が 1 の項目を探す．ここでは Nginx-service が 31000 番を使用しており，作成順番号が 1 になっている．そのため Nginx-service の Port 番号を 31000 にすることで復元を行う．復元後，Service の Port 番号が重複したタイミングと同様の形式でユーザにコマンドラインで通知を行う．図 5 の場合，Port を変更した Service 名は Nginx-service で変更後の Port 番号は 31000 番になる．

ユースケース・シナリオ

ユースケースとして東京工科大学で行っている Site Reliability Engineering の実習をあげる．ユースケースを図 6 に示す．

Nginx-Service は学生が宿題を完了するために作成した Service である．Port 番号は 31000 番に設定されている．宿題が実習開始時点までに完了しなかった学生は宿題の Pod や Service が削除できない．そのため実習課題の Service でも同じ Port 番号を使用しようとしたときに重複が発生する．ここで提案ソフトウェアが宿題の Service の Port 番号を変更することで実習課題の Service が作成できる．図 6 では Port 番号を 39999 番に変更することで 31000 番で重複が発生しないようにしている．実習終了後は学生が実習課題の Service を削除し，提案ソフトウェアが宿題の Service を復元するため，実習開始前の宿題の状態と同様の

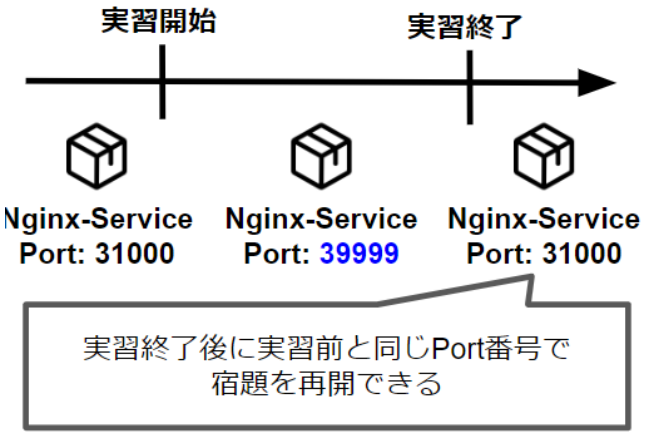

図 6: ユースケース

状態になる．これによって学生はすぐに宿題に取り組むことができる．図 6 では実習終了後に Nginx-Service の Port 番号が 31000 番に戻ることで実習開始前と同じ Port 番号で使用できる．

4. 実装

Service が作成された時の動作の実装を図 7 に示す．

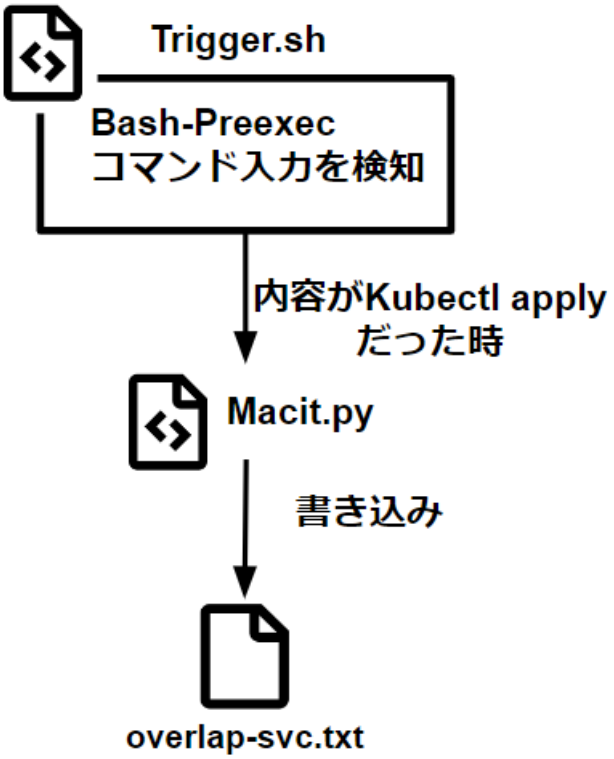

図 7: Service が作成された時の動作の実装

Triger は Bash-Preexc[*1]によってコマンドの入力を検知する．そして内容が Kubectl apply だった時，Macit を呼び出す Macit はコマンドと apply された YAML ファイルから作成された Kubernetes リソースの種類を判断し Service の時は overlap-svc というファイルに提案で示した内容を書きこむ．

次に Service の Port 番号が重複した時の動作の実装を図 8 に示す．

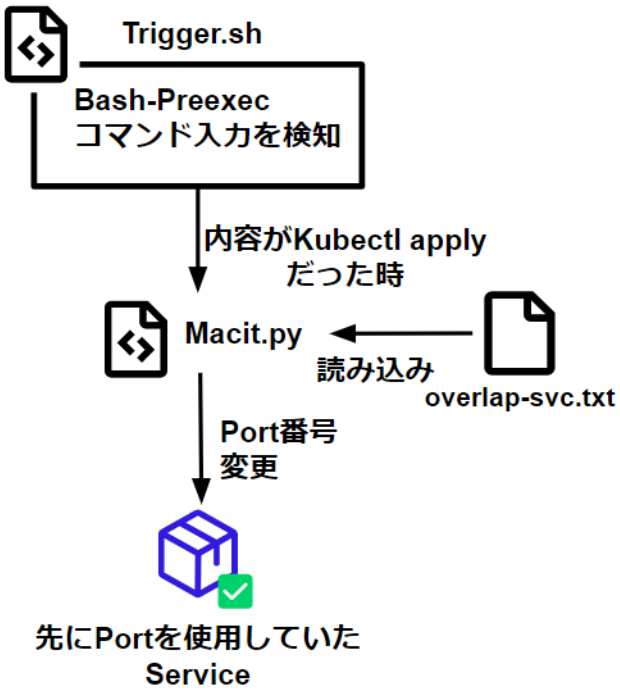

図 8: Service の Port 番号が重複した時の動作の実装

コマンドの入力検知までは Service が作成された時の動作の実装と同様である．検知後，Service の Port 番号が overlap-svc に記載されているものと重複していたら，その Service を Service 名から特定し，Port 番号の固定をやめることで，空いている Port にランダムに移動される．その後ユーザから入力されたコマンドをもう一度入力することで，Service の作成を行う．このタイミングも Service が作成された時に該当するため，手動で作成された時と同様の動作で overlap-svc に内容が書きこまれる．Service の番号が変更されたことは Service 名と変更後の Port 番号のコマンドライン出力でユーザに通知される．

最後に Service を削除した時の動作の実装を図 9 に示す．

Trigger では kubectl delete を検知する．Macit は overlap-svc から削除された Service がどの Port 番号を使用していたのかを確認し，作成順番号から，以前にその Port 番号を使用していた Service を特定する．最後にその Service の Port 番号を元の番号に復元し，削除された Service に関する行を overlap-svc から削除する．Service の番号が変更されたことは Service 名と変更後の Port 番号のコマンドライン出力でユーザに通知される．

[*1] https://github.com/rcaloras/bash-preexec

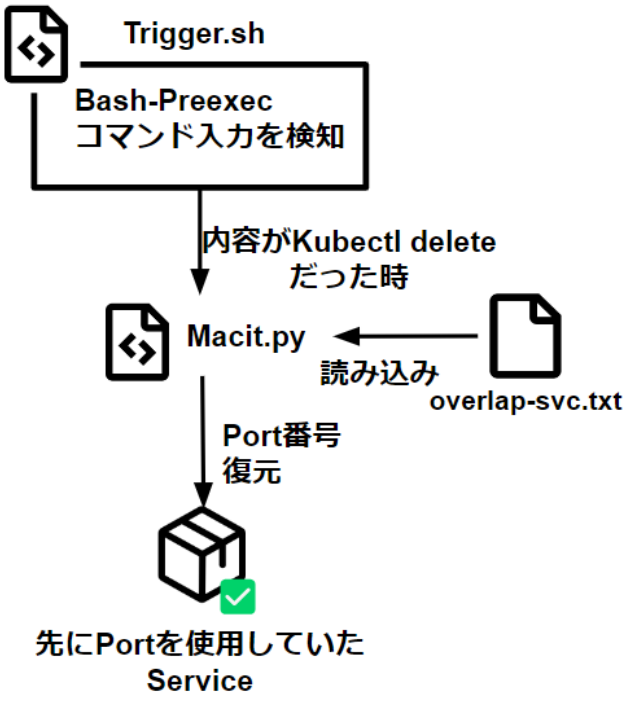

図 9: Service を削除した時の動作の実装

5. 評価実験

実験方法

実験として実習中に提案アルゴリズムで学生の Service の重複の解消を行い，その結果配置された Service が課題で求められている Service になっているかの精度で評価を行う．はじめに，Service での重複が発生したタイミングで提案ソフトウェアが動作する．次に実装したソフトウェアが動作した時に，その学生の実習課題の完了状態を完了状態が記録されているスプレッドシートから確認しその実習課題で求められている Service を資料から確認する．作成を指示されている Service と重複の解消後に実際に配置されている Service が一致しているかを確認しその正解率を精度とする．

基礎実験

基礎実験として，実際に授業 2024 年前期の実習の第 4 回から第 12 回で，全グループの Service を取得した．実習の初めの資料説明の時間中に，t サーバで kubectl コマンドを使用して，全グループのすべてのネームスペースの Service を取得した．結果はその中から学生の使用しているネームスペースで Service の種類が ClusterIP 以外のものを集計した．結果を表 1 に示す．

残っている Service が最も少なかった授業回は，第 6 回で全グループを合計して 7 個であった．最も多かった授業回は第 10 回で全グループで 39 個であった．

第 4 回から第 12 回の合計で残っている Service が最も少なかったグループはグループ 10 で 15 個であった．残っている Service が最も多かったグループはグループ 2 とグループ 4 で 29 個であった．ただし第 8 回の初めにグループ間でのメンバーの交換が行われているためそれ以前と以降では別のメンバーがグループに所属している．第 4 回から 12 回の全グループでの合計は 238 個となった．実習では各グループでそれぞれ 7 個 Port 番号が使用できる．そのため，4 人グループの場合，Service が 4 つ以上残っているとさらに実習課題の達成のため 1 人 1 つの Service を作成するため Port 番号が足りなくなってしまう．第 4 回から 12 回の全グループで，残っている Service が 4 つ以上ものは 90 件中 38 件だった．

表 1: 授業開始時に残っている Service

グループ	1	2	3	4	5	6	7	8	9	10
第 4 回	0	5	0	4	3	0	0	4	3	6
第 5 回	0	4	3	4	3	4	4	4	3	4
第 6 回	0	0	0	2	1	0	3	1	0	0
第 7 回	4	3	4	4	3	1	4	4	2	0
第 8 回	4	3	3	4	3	2	4	4	3	2
第 9 回	2	2	3	0	1	4	0	0	0	2
第 10 回	4	4	6	4	4	4	4	4	4	1
第 11 回	4	5	6	5	4	4	4	4	0	0
第 12 回	3	3	1	2	2	2	3	3	4	0

実験環境

2024 年度後期の Site Reliability Engineering の実習の受講生 40 名を対象とする．サーバの構成は前期と同様である．h サーバに提案ソフトウェアを導入し，実際の実習で使用する．実験環境を図 10 に示す．図 10 は 1 グルー

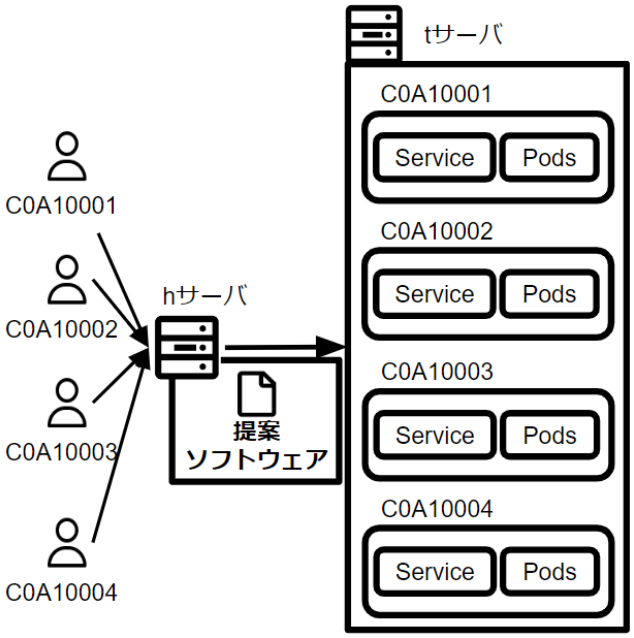

図 10: 実験環境

プの環境である．C0A10001 から C0A10004 は学生の学籍番号で，グループに所属する学生を指す．h サーバには提案ソフトウェアが配置されている．t サーバには各学生の Namespace があり，学生はそれぞれ自身の学籍番号の Namespace に実習課題を完了するための Service や Pod を配置する．実習では図 10 と同様の構成が 10 グループある．

6. 議論

本稿では 1 つの実習課題では 1 人の学生につき 1 つの Port 番号のみを使用する状況をあげた．しかし実際には複数の Service を同時に公開する必要がある実習課題もある．例えば WordPress を作成して，記事の保存に使用しているデータベースを phpMyAdmin で確認する実習課題では WordPress の Service と phpMyAdmin の Service が必要である．その場合古い Service が不要とは言えない．この問題に対して，Service に対する HTTP リクエストの履歴をもとにその Service を使用しているか判断することができる．実習は毎週行われるため，1 週間アクセスされなかった Service は実習でも宿題でも使用してないため Port 番号を変更できる．

本稿では, 1 人の学生が宿題として自分で作成した Service との Port 番号の重複を課題としてあげた．しかし実際はグループで同じサーバを使用しているため，他の学生の作成した Service と Port 番号が重複する場合もある．この場合 2 つの Service が同時に必要になるため，今回の提案は適用できない．この問題に対して，Service を別の Port 番号に移動させ，重複した番号に LoadBalancer を設置して，HTTP リクエストの送信元 IP をもとにしてそれぞれ自身が作成した Service にポートフォワーディングする方法がある．これによって，2 人の学生が同じ IP アドレスと Port 番号を使用して Service を利用してもそれぞれ自分の Service にアクセスすることができる．

本稿では復元するサービスを作成順番号が最も大きいものとした．しかし複数回分の宿題が発生した場合，学生は先に課せられた宿題を先に取り組むため，作成順番号が小さいものを復元する必要がある，この問題に対して，Service が合作成されたタイミングと，実習課題の完了状況を比べることによって，そのサービスがどの実習課題の達成のために作成されたものかを識別し，最も古い未完了の課題のための Service を復元することで学生が取り組む宿題の Service を復元できる．

7. おわりに

東京工科大学では 3 年生に向けて Site Reliability Engineering の実習を行っている．実習では，4 人で構成されたグループごとに 3 台の仮想マシンを使用する．学生は Kubernetes を操作し Service を作成して実習課題を進める．授業時間内に実習課題が完了しなかった学生はその内容を宿題として次回の授業開始時までに取り組む．課題は宿題が完了せず作成されたままになっている Service と実習課題を完了するために作成する Service で Port 番号が重複し Service の作成ができないことである．本稿では，Service に重複が発生したときに先に作成されていた Service の名前と Port 番号を保存し，Port 番号を変更することでその Port で新しく Service を作成できるようにし，Service 削除するコマンドが送信された際にファイルから以前にその Port を使用していた Service を特定し，それを復元する方法を提案した．提案方式をもとに Service 名と Port 番号の保存と，Port 番号の変更を行うソフトウェアを構築する．基礎実験では実際に第 4 回から第 12 回の実習で，実習開始時に全グループの Service を取得した．実習では各グループでそれぞれ 7 個 Port 番号が使用できる．そのため，4 人グループの場合，Service が 4 つ以上残っているとさらに実習課題の達成のため 1 人 1 つの Service を作成すると Port 番号が足りなくなってしまう．第 4 回から 12 回の全グループで，残っている Service が 4 つ以上ものは 90 件中 38 件だった．

参考文献

[1] Dickerson, M. and Chen, T.-Y.: Teaching Site Reliability Engineering as a Computer Science Elective, *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, SIGCSE 2023, New York, NY, USA, Association for Computing Machinery, p. 521–527 (online), DOI: 10.1145/3545945.3569809 (2023).

[2] ND, K. M.: Site Reliability Engineering: Application of Item Response Theory to Application Deployment Practices and Controls, *arXiv preprint arXiv:2008.06717* (2020).

[3] Burns, B., Grant, B., Oppenheimer, D., Brewer, E. and Wilkes, J.: Borg, Omega, and Kubernetes, *ACM Queue*, Vol. 14, pp. 70–93 (online), available from ⟨http://queue.acm.org/detail.cfm?id=2898444⟩ (2016).

[4] Patel, S. K., Rathod, V. and Prajapati, J. B.: Performance analysis of content management systems-joomla, drupal and wordpress, *International Journal of Computer Applications*, Vol. 21, No. 4, pp. 39–43 (2011).

[5] Reese, W.: Nginx: the high-performance web server and reverse proxy, *Linux Journal*, Vol. 2008, No. 173, p. 2 (2008).

[6] Balla, D., Simon, C. and Maliosz, M.: Adaptive scaling of Kubernetes pods, *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*, pp. 1–5 (online), DOI: 10.1109/NOMS47738.2020.9110428 (2020).

[7] Pahl, C., Brogi, A., Soldani, J. and Jamshidi, P.: Cloud Container Technologies: A State-of-the-Art Review, *IEEE Transactions on Cloud Computing*, Vol. 7, No. 3, pp. 677–692 (online), DOI: 10.1109/TCC.2017.2702586 (2019).

[8] Bernstein, D.: Containers and Cloud: From LXC to Docker to Kubernetes, *IEEE Cloud Computing*, Vol. 1, No. 3, pp. 81–84 (online), DOI: 10.1109/MCC.2014.51 (2014).

[9] Pahl, C., Brogi, A., Soldani, J. and Jamshidi, P.: Cloud

container technologies: a state-of-the-art review, *IEEE Transactions on Cloud Computing*, Vol. 7, No. 3, pp. 677–692 (2017).

[10] 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).

[11] Nguyen, N. and Kim, T.: Toward Highly Scalable Load Balancing in Kubernetes Clusters, *IEEE Communications Magazine*, Vol. 58, No. 7, pp. 78–83 (online), DOI: 10.1109/MCOM.001.1900660 (2020).

[12] Kaur, A. and Kalra, M.: Energy optimized VM placement in cloud environment, *2016 6th International Conference - Cloud System and Big Data Engineering (Confluence)*, pp. 141–145 (online), DOI: 10.1109/CONFLUENCE.2016.7508103 (2016).

[13] Ranjana, R., Radha, S. and Raja, J.: Performance study of resource aware energy efficient VM placement algorithm, *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, pp. 86–89 (online), DOI: 10.1109/WiSPNET.2016.7566096 (2016).

[14] Nguyen, N. T. and Kim, Y.: A Design of Resource Allocation Structure for Multi-Tenant Services in Kubernetes Cluster, *2022 27th Asia Pacific Conference on Communications (APCC)*, pp. 651–654 (online), DOI: 10.1109/APCC55198.2022.9943782 (2022).

[15] Ylonen, T., Turner, P., Scarfone, K. and Souppaya, M.: Security of interactive and automated access management using Secure Shell (SSH), *NISTIR 7966, National Institute of Standards and Technology* (2015).

[16] Al-Ajlan, A. and Zedan, H.: Why moodle, *2008 12th IEEE International Workshop on Future Trends of Distributed Computing Systems*, IEEE, pp. 58–64 (2008).

[17] Alves, G. R., Viegas, M. C., Marques, M. A., Costa-Lobo, M. C., Silva, A. A., Formanski, F. and Silva, J. B.: Student performance analysis under different moodle course designs, *2012 15th International Conference on Interactive Collaborative Learning (ICL)*, pp. 1–5 (online), DOI: 10.1109/ICL.2012.6402181 (2012).

[18] Wang, Z., Liu, H., Han, L., Huang, L. and Wang, K.: Research and Implementation of Scheduling Strategy in Kubernetes for Computer Science Laboratory in Universities, *Information*, Vol. 12, No. 1 (online), DOI: 10.3390/info12010016 (2021).

[19] 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).

[20] Jouin, L.: Network service mesh solving cloud native IMS networking needs (2020).

