生成AIでのYAMLファイルの記述ミスの作成とパターンマッチングによるKubernetes Podのエラー原因の検出率向上

手塚 雄星[1]　小山 智之[2]　串田 高幸[1]

概要：東京工科大学クラウド・分散システム研究室では，Kubernetes の CronJob とシェルスクリプトを組み合わせて Prometheus のメトリクスデータを定期的にバックアップをするための開発作業を行っている．開発時に YAML ファイルやスクリプトの記述ミスがあると，それらを修正し変更の適用を繰り返す必要があり，エラーの原因箇所の特定に時間がかかっている．提案方式では，エラーの原因特定にかかる時間を短縮するために，kubectl describe コマンドの結果とルールのパターンマッチングにより YAML ファイルのエラーの原因箇所を特定する．研究室では，Prometheus のメトリクスデータをバックアップするために開発作業で YAML ファイルが作成されている．ルールの作成では，生成 AI によって開発作業で作成した記述ミスを含まない YAML ファイルから記述ミスを想定したルールを作成する．ルールには，ルール名，キーワードリスト，エラーの原因箇所で構成されている．キーワードリストには，kubectl describe コマンドの結果の Events に含まれる Message から取り出した単語がカンマ区切りで含まれる．エラーの原因箇所には YAML ファイルの行番号と行の内容が含まれる．生成 AI のプロンプトには，バックアップを行うことができる開発作業で作成された記述ミスを含まない 6 個の YAML ファイル，生成 AI に対しての指示，ルールの形式，開発作業で作成された記述ミスを含む YAML ファイルを指定した．開発作業で作成された記述ミスを含む YAML ファイルと kubectl describe コマンドの結果をルールと照合し，ルールに合致した箇所をエラーの原因箇所と判断し出力する．評価実験では，12 個の開発作業で作成された記述ミスを含む YAML ファイルを対象に，提案ソフトウェアが検出した割合を検出率で評価した．検出率は，12 個のうち 5 個検出されたため，約 41.7%であった．

1. はじめに

背景

Kubernetes は，分散クラスタ上でコンテナ化されたアプリケーションのデプロイメント，管理，スケーリングを自動化するためのオープンソースのコンテナオーケストレーションシステムである [1]．Google の Borg プロジェクトを起源として，2014 年に公開されてからクラウドネイティブなアーキテクチャの中核技術として広く普及している．軽量で柔軟なコンテナ技術を活用して，マイクロサービスベースのアプリケーションをクラスタ内のノードに配置するためのスケジューリング機能を備えている．kube-scheduler は，ユーザーが定義したリソース要求やノードアフィニティのユーザー仕様にもとづいて，最適なワーカーノードを選定して，Pod を配置する役割を担っている．QoS を考慮したフィルタリングやランキング処理を通じて，CPU やメモリの使用率，ネットワーク状態のシステムコンテキストに応じた動的なスケジューリングを実現している．オートスケーリング，ロードバランシング，監視，障害耐性の機能が統合されていて，Container as a Service の基盤としても活用されている．

コンテナは，クラウドコンピューティング環境において，アプリケーションと依存関係を 1 つの単位としてパッケージ化して実行するための技術である [2]．仮想マシンよりも軽量で起動時間が短く，リソースの消費も少ないため，スケーラビリティと可搬性に優れている．Kubernetes のようなコンテナオーケストレーションシステムにより，コンテナはクラスタ内の複数ノードに分散配置されて，リソース使用率やノードの状態に応じて動的にスケジュールされる．システム全体の負荷分散が最適化されて，アプリケーションの可用性とパフォーマンスが向上する．また，クラウドインフラの状態やユーザーの要求に応じて，実行中の

[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

コンテナ数やイメージ転送時間の要素を考慮して配置されるため，運用効率の向上とコスト削減に寄与する．

コンテナやマイクロサービスの持つ高い移植性と拡張性は，ソフトウェア開発のスピードと柔軟性を大きく向上させた [3]．構成ミスがシステムに紛れ込むリスクも高まっている．まず，ベースイメージに含まれる設定ミスが修正されずに後続のイメージに継承することがある．次に，オーケストレーション環境では複数のソフトウェアバージョンが並行して運用されるため，異なる設定やチューニングが必要となり，コンテナごとに継続的な設定管理が求められる．さらに，ソフトウェアのバージョンだけでなく，リソースの割り当てやシステムアーキテクチャ，デプロイメント環境全体を考慮した設定が不可欠である．それにより，複雑性が従来よりも設定ミスの発生を助長する要因となっている．

YAML（Yet Another Markup Language）は文章の整形よりもデータの表現に重点を置いており，Perl，C，XML，HTML の言語から着想を得て設計されている [4]．そのため，構文は直感的でありながら複雑な構成情報の記述にも対応が可能である．YAML は JSON の上位互換であり，JSON ファイルとの互換性を持つため，既存のシステムとの統合が容易である．さらに，クエリの簡素化において，JSON や XML よりも優れており，テンプレート用途においては利便性がある．YAML テンプレートは設定ファイルの標準化や保守性の向上に寄与しており，セキュリティイベントの抽出やログ解析にも活用されている．

データセットにおける障害のうち，33 件は設定ミス（人為的ミス）に起因している [5]．10 件は Kubernetes クラスタにおけるノードとサービスのリソースサイジングの不備が原因で発生している．サービスのリソースが不足するとアプリケーションが故障することやリソースが過剰になるとノードが故障することがある．Kubernetes における過負荷によるリソース枯渇の問題が顕在化している．具体的な内訳としては Kubernetes（K8s）の設定ミスが 19 件，プラグインの設定ミスが 30 件，外部ソフトウェアの設定ミスが 11 件と分類されており，設定ミスが Kubernetes クラスタの安定性に影響を与えている．Kubernetes 環境における構成管理の精度が障害回避において重要である．

Microsoft Teams の高重大度インシデント 152 件を分析した結果，インシデントの根本原因は広範囲に及ぶ [6]．最も多いのがコードバグで，全体の約 27%を占める．なぜなら，開発段階でのロジックミスや不具合がサービス障害につながるケースが多いからである．次に多いのが依存サービスの障害で 16.4%である．Teams は外部のクラウドサービスや API，認証基盤に依存しており，障害が影響を及ぼすリスクがある．インフラ関連の問題は 15.8%である．サーバーやストレージ，ネットワーク機器の物理または仮想インフラの不具合が原因になるケースが含まれる．デプロイミスは 13.2%である．変更のリリース時に適切な検証が行われないことや誤ったバージョンが展開されたことによる障害がある．設定ミス（コンフィグバグ）は 12.5%である．環境変数やフラグの誤設定，アクセス権限の不備，運用上の細かなミスが影響を与える．データベースやネットワーク障害は 10.5%である．クエリの遅延や接続断，パフォーマンス劣化が原因となる．認証エラーは 4.6%である．ユーザーのログインやサービス間の認証処理における不具合が含まれている．したがって，コードや設定に起因するインシデントはコードバグの約 27%と設定ミスの 12.5%の 2 つある．27 と 12.5 の和が 39.5 になるため，約 40%で開発や構成段階で品質管理が重要である．しかし，インフラや依存関係，運用ミスにあたる非コード系の要因である依存サービスの障害で 16.4%とインフラ関連の問題の 15.8%とデプロイミスの 13.2%とデータベースやネットワーク障害の 10.5%の 4 つがある．16.4 と 15.8 と 13.2 と 10.5 と 4.6 の和が 60.5 になるため，約 60%を占めており，システム全体の設計や運用体制の堅牢性も重要である．

オンラインサービスにおける障害のうち 50.4%が誤った変更に起因している [7]．具体的には，コードの欠陥，設定ミス，リソースの競合，ソフトウェアバージョンの不整合である．誤った変更はシステムの挙動に影響を与えて，サービスの停止や性能低下を引き起こす要因となる．

東京工科大学にはクラウド・分散システム研究室（以後，CDSL とする）がある．CDSL では Prometheus のメトリクスデータを定期的にバックアップするために Kubernetes の CronJob リソースを活用している．開発者として佐藤健斗さんが開発作業を行っている（以降，佐藤さんとする）．図 1は佐藤さんによる Prometheus をバックアップする開発作業を表している．図 1では，(1) で佐藤さんは Kubernetes のリソース定義を YAML ファイルに記述する．開発作業で作成された YAML ファイルには CronJob リソースの設定が含まれており，バックアップ処理を定期的に実行するためのスケジュールや使用するコンテナ，実行するコマンドが定義されている．(2) で kubectl apply コマンドを使用して開発作業で作成された YAML ファイルを Kubernetes クラスタに適用する．CronJob がクラスタ上に登録されて，指定されたスケジュールに従って処理が自動的に開始する．(3) で CronJob がスケジュールに従って起動すると，Kubernetes は Job リソースを生成する．Job は一度限りの処理を実行するためのリソースであり，Pod が作成される．Pod は定義したコンテナイメージから起動されてバックアップ処理が実行される．(4) で Pod 内では，事前に用意したシェルスクリプトが実行される．具体的には，Prometheus の PushGateway からメトリクスデータを取得する処理や，取得したデータを保存する処理，保存先である NAS（Network Attached Storage）への転送処理である．Job はシェルスクリプトに対して実行指

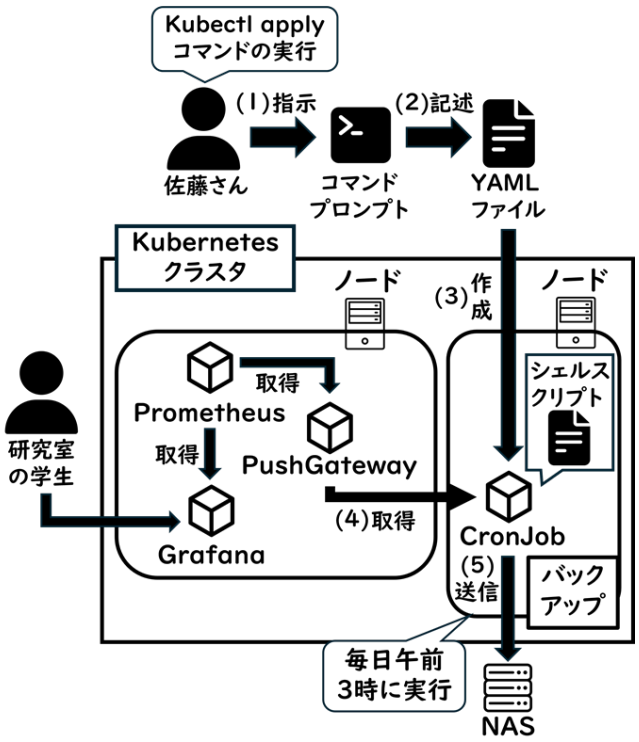

図 1 佐藤さんによる Prometheus をバックアップする開発作業

示を出して，シェルスクリプトが順に処理を進めることでバックアップが完了する．PushGateway は，Prometheus において短期的なジョブやバッチ処理がメトリクスを一時的に保存するためのコンポーネントである．バックアップ処理では，PushGateway のエンドポイントからメトリクスデータを取得する．取得には HTTP リクエストを用いて，PushGateway が保持している最新のメトリクス情報をテキスト形式で取得する．(5) で取得したメトリクスデータは NAS に送信される．NAS はネットワーク経由でアクセス可能なストレージであり，バックアップデータの長期保存や共有に適している．転送は安全な通信手段が用いられており，データの整合性と安全性が確保される．CDSL では Kubernetes の CronJob とシェルスクリプトを組み合わせることで，Prometheus のメトリクスデータを定期的かつ自動的にバックアップする仕組みを構築している．

開発作業で作成された YAML ファイル

プログラム 1は開発作業で作成された YAML ファイルを表している．プログラム 1では，1 行目の apiVersion は batch/v1 で CronJob リソースの API バージョンを指定している．2 行目の kind は CronJob で定義する Kubernetes リソースの種類が CronJob であることを示している．3 行目の metadata は CronJob のメタデータセクションの開始を示している．4 行目の name は prometheus-backup で CronJob の名前を指定している．5 行目の spec は CronJob の仕様を定義するセクションの開始を示している．6 行目

プログラム 1　開発作業で作成された YAML ファイル

```
1  apiVersion: batch/v1
2  kind: CronJob
3  metadata:
4    name: prometheus-backup
5  spec:
6    #schedule: "0 3 * * *"
7    schedule: "*/1 * * * *"
8    jobTemplate:
9      spec:
10       template:
11         spec:
12           containers:
13           - name: prometheus-backup
14             image: c0a22169d8/backup-prometheus
                :latest
15             args:
16             - "/home/monitoring/prometheus-
                backup/prometheus-backup.sh"
17           restartPolicy: OnFailure
```

の#schedule は”0 3 * * *”は，コメントアウトされたスケジュールで毎日 3 時に実行する設定の例である．7 行目の schedule は”*/1 * * * *”で 1 分ごとにジョブを実行する Cron 形式のスケジュールである．8 行目の jobTemplate は実行されるジョブのテンプレートを定義するセクションの開始を示している．9 行目の spec は jobTemplate 内のジョブ仕様の開始を示している．10 行目の template は Pod テンプレートの定義を開始するフィールドである．11 行目の spec は Pod の仕様を定義するセクションの開始を示している．12 行目の containers は Pod 内で実行されるコンテナのリストを定義するフィールドである．13 行目の- name は prometheus-backup でコンテナの名前を指定している．14 行目の image は c0a22169d8/backup-prometheus:latest で使用する Docker イメージを指定している．15 行目の args はコンテナ起動時に渡す引数のリストを定義するフィールドである．16 行目の- ”/home/monitoring/prometheus-backup/prometheus-backup.sh”は，コンテナ起動時に実行するシェルスクリプトのパスである．17 行目の restartPolicy は OnFailure でジョブが失敗した場合にのみ再起動するポリシーを指定している．

開発作業で出力された kubectl describe コマンドの結果

プログラム 2は開発作業で出力された kubectl describe コマンドの結果を表している．プログラム 2では，1 行目の Status は，Running で Pod が稼働中の状態であることを示している．2 行目の Events は，この Pod に関連するイベントログの一覧が始まることを示している．3 行目の Type Reason Age From Message は，イベントログの各列のヘッダである．イベントの種類，理由，経過時間，発生元，詳細メッセージを表している．5 行目

プログラム 2　開発作業で出力された kubectl describe コマンドの結果

```
1  Status: Running
2  Events:
3    Type Reason Age From Message
4    ---- ------ ---- ---- -------
5    Normal Scheduled 117s default-scheduler
     Successfully assigned monitoring/prometheus-
     backup-29187618-54d9v to monitoring-worker1
6    Warning Failed 70s (x4 over 113s) kubelet
     Error: failed to create containerd task:
     failed to create shim task: OCI runtime
     create failed: runc create failed: unable to
      start container process: exec: "/home/
     monitoring/prometheus-backup/prometheus-
     backup.sh": stat /home/monitoring/prometheus
     -backup/prometheus-backup.sh: no such file
     or directory: unknown
7    Normal Pulled 70s kubelet Successfully
     pulled image "c0a22169d8/backup-prometheus:
     latest" in 1.294s (1.294s including waiting
     ). Image size: 3643355 bytes.
8    Warning BackOff 43s (x7 over 110s) kubelet
     Back-off restarting failed container
     prometheus-backup in pod prometheus-backup
     -29187618-54d9v_monitoring(95534006-402a
     -4285-8e55-8dd9491195c1)
```

の Normal Scheduled 117s default-scheduler Successfully assigned monitoring/prometheus-backup-29187618-54d9v to monitoring-worker1 は，Pod が 117 秒前に monitoring-worker1 に正常に割り当てられたことを示すイベントである．6 行目の Warning Failed 70s (x4 over 113s) kubelet Error: failed to create containerd task: ... は，コンテナ起動時に/home/monitoring/prometheus-backup/prometheus-backup.sh が見つからず，起動に失敗したことを示す警告イベントである．113 秒の間に 4 回発生している．7 行目の Normal Pulled 70s kubelet Successfully pulled image ”c0a22169d8/backup-prometheus:latest” ... は，指定された Docker イメージが 70 秒前に正常に取得されたことを示すイベントである．イメージサイズは約 3.6MB である．8 行目の Warning BackOff 43s (x7 over 110s) kubelet Back-off restarting failed container prometheus-backup ... は，コンテナの起動失敗が繰り返されたので，43 秒前から再起動を一時停止（BackOff）していることを示す警告イベントである．110 秒の間に 7 回発生している．

田中美帆らの提案方式では，Pod の Status や kubectl describe コマンドの Events，起動に成功した場合の開発作業で作成された記述ミスを含まない YAML ファイル (以降，記述ミスのない YAML ファイル) と起動に失敗した場合の 開発作業で作成された記述ミスを含む YAML ファイル (以降，記述ミスのある YAML ファイル) の差分で 3 つの要素をもとにルールが保存されたファイルを作成して，エラーの原因箇所を特定する仕組みを構築している [8].

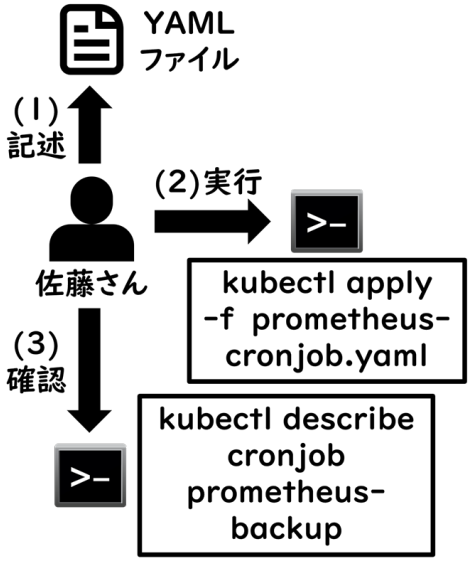

図 2　佐藤さんが開発作業で繰り返した手順

課題

課題は，Kubernetes クラスタ内での開発作業において Pod を起動する際に，下記の手順を繰り返すことでエラーの原因箇所を特定することに時間がかかっていることである．

(1) YAML ファイルやシェルスクリプトの記述
(2) kubectl apply -f prometheus-cronjob.yaml の実行
(3) kubectl describe cronjob prometheus-backup の実行

図 2は佐藤さんが開発作業で繰り返した手順を示している．図 2では，佐藤さん，YAML ファイル，シェルスクリプト，kubectl apply コマンド，kubectl describe コマンドで構成される．佐藤さんは CDSL の佐藤健斗さんをあらわす．YAML ファイルは開発作業で作成されたファイルをあらわす．シェルスクリプトはバックアップの開発作業において記述されたスクリプトをあらわす．kubectl apply コマンドは Kubernetes クラスタで Pod を起動するための実行コマンドをあらわす．kubectl describe コマンドは Kubernetes クラスタで Pod を起動した後の状況を確認するための実行コマンドをあらわす．(1) では，YAML ファイルとシェルスクリプトを記述している．これは，開発作業において作成したファイルとスクリプトである．(2) では，kubectl apply -f prometheus-cronjob.yaml で kubectl apply コマンドを実行している．これは kubectl apply コマンドで CronJob リソースの Pod を起動させている．(3) では，kubectl describe cronjob prometheus-backup で kubectl describe コマンドを実行している．これは kubectl describe コマンドで Pod の状態を確認している．

基礎実験

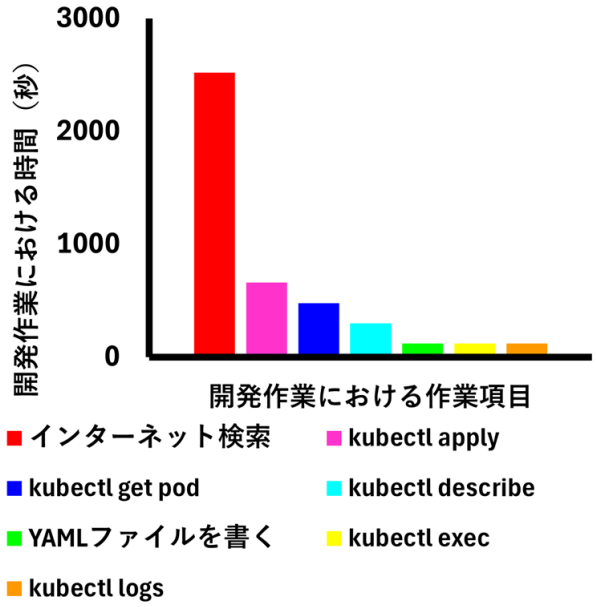

図 3　2025 年 9 月 22 日の 15:04 から 16:03 に佐藤さんが開発作業をした内訳

図 3は 2025 年 9 月 22 日の 15:04 から 16:03 に佐藤さんが開発作業をした内訳を棒グラフで表している．図 3では，X 軸で開発作業における作業項目を表している．Y 軸で開発作業における時間を表しており，単位は秒をとる．開発作業における時間は，開発作業における作業項目の各々にかかる時間を示している．凡例の赤色はインターネット検索を表す．ピンク色は kubectl apply を表す．青色は kubectl get pod を表す．水色は kubectl describe を表す．緑色は YAML ファイルを書くを表す．黄色は kubectl exec を表す．オレンジ色は kubectl logs を表す．インターネット検索が 2520 秒で，kubectl apply が 660 秒で，kubectl get pod が 480 秒で，kubectl describe が 300 秒で，YAML ファイルを書くが 120 秒で，kubectl exec が 120 秒で，kubectl logs が 120 秒である．最も時間がかかっている項目は，インターネット検索である．これは，エラーの原因箇所を特定するために要する時間である．kubectl get pod，kubectl describe，kubectl logs の 3 つのコマンドで Pod の状況を確認している時間の合計は 480 + 300 + 120 = 900 であるため，900 秒である．900 秒は kubectl apply，kubectl get pod，kubectl describe，YAML ファイルを書く，kubectl exec，kubectl logs の時間の合計のうち 50%を占めている．

各章の概要

第 2 章では，関連研究について説明する．第 3 章では，課題について解決するための提案方式について説明する．第 4 章では，提案した手法の実装について説明する．第 5 章では, 評価実験として実験内容と実験結果と分析について説明する．第 6 章では，提案方式についての議論を説明する．最後に，第 7 章にて結論を説明する．

2. 関連研究

YAML ファイルをモデルとしてノード単位での解析や検証を可能にする技術的フレームワークを提案する研究がある [9]．この研究では，EMC YAML Driver と Epsilon Validation Language（EVL）を用いることで，YAML ファイル内の構造的な誤りや設定値の不整合を検出して，Pod が起動に失敗する原因となる箇所を特定や提示している．YAML ファイルを抽象構文木としてスカラー，マッピング，リストノードを対象に検索や編集，検証を行うことで，Kubernetes のリソース定義におけるエラーの早期発見と修正を支援している．一方で，YAML 構造の抽象化や検証ルールの記述に関する汎用的なフレームワークの整備と Kubernetes のリソース仕様に対応した検証メカニズムの拡張に改善の余地がある．

Docker や Kubernetes のクラウド仮想環境でデプロイされたコンテナに対して，誤設定を検出するモデルを提案する研究がある [10]．この研究では，Docker や Kubernetes を用いたコンテナデプロイメントにおける誤設定の検出や分析，対策を扱っている．コンテナ技術はソフトウェア開発ライフサイクルを可能にしているが，デプロイメントの初期段階での誤設定により，脆弱な環境を生み出すセキュリティ上の懸念がある．誤設定を分析することで，悪用される前に回避するための防御を提供している．一方で，対象とするコンテナデプロイメントの誤設定がセキュリティに限定されているため，ユースケースの誤設定を検出できない．

分散クラウド環境における誤設定の防止とエラー原因検出のメカニズムを提案する研究がある [11]．この研究では，スキーマベースの検証によって，スキーマの条件を満たさない構成を防ぐことで，間違いを含む構成のリスクを軽減している．ユーザーは YAML 形式でスキーマを定義して，システムは内部的に JSON 形式に変換して検証を行う．検証は，新しい構成がシステムに適用される前に実行される．検証をしている際にエラーが検出された場合には，新しい構成がシステムに適用されない．一方で，検証においてスキーマの比較をしているため，ユースケースのシェルスクリプトのパスの誤設定を検出できない．

3. 提案

提案方式

提案方式では，エラーの原因特定にかかる時間を短縮するために，ルールを用いたパターンマッチングにより YAML ファイルのエラーの原因箇所を特定する．図 4は，提案方式を示している．図 4では，プロンプト，生成 AI，記述ミスを追加した YAML ファイル，Kubernetes クラスタ，

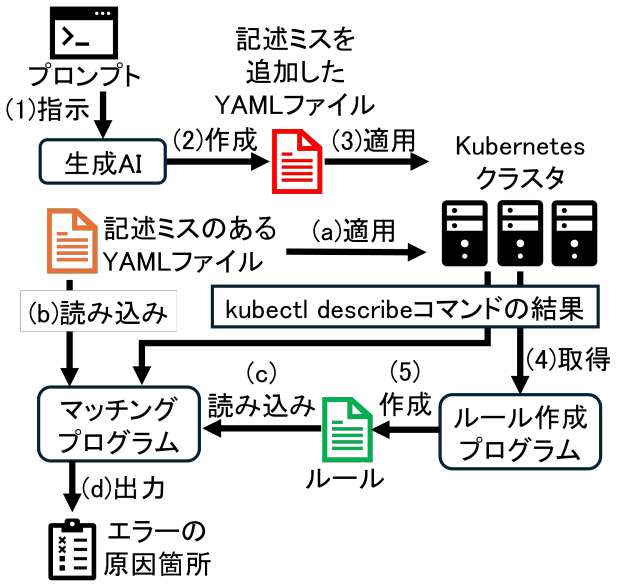

図 4　提案方式

kubectl describe コマンドの結果，ルール作成プログラム，ルール，マッチングプログラム，記述ミスのある YAML ファイル，エラーの原因箇所で構成されている．プロンプトは，生成 AI に何をしてもらうのか指示をする文章である．生成 AI は，大規模言語モデルをもとにして出現確率を計算することでテキストを生成する技術である．開発作業において発生するエラーに対して，最大限に対策するため，生成 AI を使用してルールを生成する．記述ミスを追加した YAML ファイルは，プロンプトの指示を受けて生成 AI が作成した YAML ファイルである．Kubernetes クラスタは，3 つのノードで構成される．kubectl describe コマンドの結果は，記述ミスを追加した YAML ファイルと記述ミスのある YAML ファイルを適用した後にコマンドを実行した際の結果である．ルール作成プログラムは，kubectl describe コマンドの結果からルールを作成するためのプログラムである．ルールはルール名やキーワードリストが含まれている．マッチングプログラムは，ルールと記述ミスのある YAML ファイルと kubectl describe の結果を比較して一致するか確認するプログラムである．記述ミスのある YAML ファイルは，開発作業で作成された記述ミスを含む YAML ファイルである．エラーの原因箇所は，ルールと記述ミスのある YAML ファイル，kubectl describe コマンドの結果を比較した後に出力される．エラーの原因箇所には YAML ファイルの行番号と行の内容が含まれる．

(1) から (5) では，ルールを作成するまでの過程を示している．(1) でプロンプトで生成 AI に指示する．(2) で生成 AI で記述ミスを追加した YAML ファイルを作成する．(3) で記述ミスを追加した YAML ファイルで Kubernetes クラスタに適用する．(4) でルール作成プログラムが kubectl describe コマンドの結果を取得する．(5) でルール作成プログラムでルールを作成する．(a) から (d) では，エラーの原因箇所が出力されるまでの過程を示している．(a) で記述ミスのある YAML ファイルを Kubernetes クラスタで適用して kubectl describe コマンドの結果を取得する．(b) で記述ミスのある YAML ファイルをマッチングプログラムに読み込む．(c) でルールをマッチングプログラムに読み込む．(d) でマッチングプログラムからエラーの原因箇所を出力する．

プロンプト

プログラム 3は生成 AI に指示するプロンプトを表している．プログラム 3では，1 行目から 7 行目は，生成 AI にしてもらうタスクをあらわす．具体的には，記述ミスのない YAML ファイルを参考にして，指定した項目を含めて記述ミスのある YAML ファイルを作成してもらうタスクである．9 行目は，YAML ファイルの説明をあらわす．10 行目は，YAML ファイルの記述の開始位置をあらわす．11 行目は，YAML ファイルを記述した年月日と日時をあらわす．12 行目から 44 行目は，記述ミスのない YAML ファイルをあらわす．YAML ファイルは研究室の DNS のバックアップ作業で記述して問題なく Pod が起動したものである．この YAML ファイルで 1 分おきにバックアップできる．

記述ミスを追加した YAML ファイル

プログラム 4は記述ミスを追加した YAML ファイルを表している．プログラム 4では，1 行目の apiVersion は，batch/v1 で CronJob リソースの API バージョンを表している．2 行目の kind は，定義する Kubernetes リソースの種類が CronJob であることを表している．3 行目の metadata は，CronJob のメタデータセクションの開始を表している．4 行目から 5 行目は，名前が dns-backup でネームスペースが default であることを表している．6 行目の spec は，CronJob の仕様を定義するセクションの開始を表している．7 行目の schedule: ”*/1 * * * *”は，1 分ごとにジョブを実行する Cron 形式のスケジュールである．8 行目の jobTemplate は，ジョブの雛形である．主な内容は Pod テンプレートとなる．9 行目の spec は，ジョブの仕様を記述するセクションの開始を表している．10 行目は起動する Pod の雛形を記述する Pod テンプレートを表している．11 行目の spec は，ジョブの仕様を記述するセクションの開始を表している．12 行目から 15 行目は，コンテナの名前，イメージ名，コマンドを表している．15 行目の command は，コンテナが起動時に渡される引数を表している．この引数には，記述ミスをしたスクリプトが設定されている．15 行目のスクリプトにおける正解の記述は，command: [’bash’, ’/backup.sh’] である．それに対して，実際には/backup.shh と設定されている．この記述ミスに

プログラム 3　生成 AI に指示するプロンプト

```
タスク
開発作業で作成された記述ミスを含まない YAML ファイルから記述ミスを含む YAML ファイルを 20 個作成してください．
YAML ファイルの構成は変えないでください．
YAML ファイルごとに記述ミスは必ず一箇所で作成してください．
作成した記述ミスを含まない YAML ファイルにコメントで正誤を記述してください．

以下は開発作業で作成された記述ミスを含まない YAML ファイルです．
YAML ファイル
「2025-09-12 13:00~15:00」
apiVersion: batch/v1
kind: CronJob
metadata:
 name: dns-backup
 namespace: default
spec:
 schedule: "*/1 * * * *"
 jobTemplate:
   spec:
     template:
       spec:
         containers:
         - name: dns-backup
           image: c0a22100/dns-backup:latest
           command: ['bash', '/backup.sh']
           env:
           - name: DB_HOST
             value: 192.168.100.35:30104
           - name: DB_USER
             value: cdsl
           - name: DB_PASS
             value: cdsl
           - name: DB_NAME
             value: powerdns
           volumeMounts:
           - name: backup-volume
             mountPath: /backup
         volumes:
         - name: backup-volume
           hostPath:
             path: /data
             type: DirectoryOrCreate
         restartPolicy: OnFailure
```

プログラム 4　記述ミスを追加した YAML ファイル

```
apiVersion: batch/v1
kind: CronJob
metadata:
 name: dns-backup
 namespace: default
spec:
 schedule: "*/1 * * * *"
 jobTemplate:
   spec:
     template:
       spec:
         containers:
         - name: dns-backup
           image: c0a22100/dns-backup:latest
           command: ['bash', '/backup.shh'] # 誤: /backup.shh / 正: /backup.sh
           env:
           - name: DB_HOST
             value: 192.168.100.35:30104
           - name: DB_USER
             value: cdsl
           - name: DB_PASS
             value: cdsl
           - name: DB_NAME
             value: powerdns
           volumeMounts:
           - name: backup-volume
             mountPath: /backup
         volumes:
         - name: backup-volume
           hostPath:
             path: /data
             type: DirectoryOrCreate
         restartPolicy: OnFailure
```

より Pod を起動した際にエラーが生じている．16 行目から 24 行目は，環境変数を表している．25 行目から 27 行目の volumeMounts は，コンテナ内部にホスト側のディレクトリをマウントする設定を表している．名前は backup-volume で，コンテナ内の/backup ディレクトリにマウントされる．28 行目から 29 行目の volumes は，ホスト側のディレクトリを指定する設定を表している．backup-volume という名前のボリュームが，ホストの/data ディレクトリを参照するように定義されている．30 行目の hostPath は，ホストのファイルシステム上のパスをコンテナにマウントするための設定を表している．/data ディレクトリを指定している．31 行目の path: /data は，ホスト側のディレクトリパスを表している．32 行目の type: DirectoryOrCreate は，指定したディレクトリが存在しない場合に作成することを表している．バックアップ先ディレクトリが存在する状態で Pod が起動できる．33 行目の restartPolicy: OnFailure は，ジョブの Pod が失敗した場合に再起動するポリシーを表している．成功時には再起動せず，失敗時のみ再試行される．

ルール作成プログラム

ルール作成プログラムは，生成 AI が作成した記述ミスを追加した YAML ファイルを適用した際に発生するエラーからルールを作成する．入力は，記述ミスを追加した

YAML ファイルで作成されたリソースに対しての kubectl describe コマンドの結果である．出力はルールである．

プログラム 5は kubectl describe コマンドの結果における Events を示している．プログラム 5では，Events の Type は Warning であり，Reason は BackOff である．Message は，Pod に含まれる dns-backup コンテナが正常に起動できずに，kubelet が再起動を試みているが失敗が続いているため，再起動の待機を表している．そのため，Pod を起動する際のエラーが kubectl describe コマンドの結果における Events の Message に出力される．

プログラム 5 kubectl describe コマンドの結果における Events

```
Events:
  Type Reason Age From Message
  ---- ------ ---- ---- -------
  Warning BackOff 7s (x25 over 5m18s) kubelet
  Back-off restarting failed container dns-
  backup in pod dns-backup-29422560-
  f6zfr_default(f09e6188-0a07-47d1-b6fc-
  d4a78309eeda)
```

プログラム 6はキーワードリストを示している．キーワードリストは，ルールに含まれる項目の 1 つである．マッチングプログラムで記述ミスのある YAML ファイルの kubectl describe の結果とルールを比較するために設定している．プログラム 6では，プログラム 5の kubectl describe コマンドの結果における Events を JSON 形式で取得するために kubectl get events コマンドで Events の Message だけを抽出している．そして，Message に含まれるダブルクォーテーションで囲まれている単語は変数に代入しておく．その後，抽出した Message の半角スペースをカンマに置き換えて単語ごとにダブルクォーテーションで囲むようにする．

プログラム 6 キーワードリスト

```
["Back-off","restarting","failed","container
","dns-backup","in","pod","dns-backup
-29422560-f6zfr_default(f09e6188-0a07-47d1-
b6fc-d4a78309eeda)"]
```

プログラム 7はルールの例である．プログラム 7のルールの例では，JSON 形式で 2 行目は Key を name として値を Back-off restarting failed container とする．Back-off restarting failed container は，kubectl describe コマンドの結果における Events の Message の初めから 4 単語までを抽出したものである．3 行目は Key を keyword_list として値を [”Back-off”,”restarting”,”failed”,”container”,”dns-backup”,”in”,”pod”,”dns-backup-29422560-f6zfr_default (f09e6188-0a07-47d1-b6fc-d4a78309eeda)”] とする．単語が 10 個以上ある場合には，先頭の 10 単語を使用する．この keyword_list はマッチングプログラムに使用するキーワードリストを表す．4 行目は Key を parameter として値を”spec.jobTemplate.spec.template.spec. containers.command”とする．parameter は，マッチングプログラムで記述ミスのある YAML ファイルにおけるエラーの原因箇所を探すために設定している．生成 AI が作成した記述ミスを追加した YAML ファイルに含まれるコメントを含む行を誤りのある行として取り出す．例えば，プログラム 4では誤りのある行は，「♯ 誤: /backup.shh / 正: /backup.sh」である．取り出した行から Key を取得し，YAML ファイルの構造を解析し最上位からその Key までの階層構造をドットでつないで表現する．例えば，”spec.jobTemplate”は spec の配下にある jobTemplate であることをあらわす．

プログラム 7 ルールの例

```
{
"name": "Back-off restarting failed container
",
"keyword_list": ["Back-off","restarting","
failed","container","dns-backup","in","pod","
dns-backup-29422560-f6zfr_default(f09e6188-0
a07-47d1-b6fc-d4a78309eeda)"],
"parameter": "spec.jobTemplate.spec.template.
spec.containers.command"
}
```

マッチングプログラム

マッチングプログラムは，ルールと kubectl describe コマンドの結果を比較して一致するか確認して，記述ミスのある YAML ファイルに該当する箇所があるか調べる役割がある．入力は，記述ミスのある YAML ファイル，kubectl describe コマンドの結果，ルールである．出力は，エラーの原因箇所である．マッチングプログラムでは次の順序で処理を行う．

(1) 記述ミスのある YAML ファイルを入力として読み込む．
(2) kubectl apply コマンドを実行する
(3) kubectl describe コマンドの結果を取得する
(4) ルールを読み込む
(5) ルールと kubectl describe の結果をマッチングする
(6) エラーの原因箇所が出力される

プログラム 8は，記述ミスのある YAML ファイルである．プログラム 8では，1 行目の apiVersion は batch/v1 で CronJob リソースの API バージョンを指定している．2 行目の kind は CronJob で定義する Kubernetes リソースの種類が CronJob であることを示している．3 行目の metadata は CronJob のメタデータセクションの開始を示してい

る．4 行目の name は postgresql-backup で CronJob の名前を指定している．5 行目の namespace は ssh-monitoring で，この CronJob が属する Namespace を指定している．6 行目の spec は CronJob の仕様を定義するセクションの開始を示している．9 行目の jobTemplate は実行されるジョブのテンプレートを定義するセクションの開始を示している．10 行目の spec は jobTemplate 内のジョブ仕様の開始を示している．11 行目の template は Pod テンプレートの定義を開始するフィールドである．12 行目の spec は Pod の仕様を定義するセクションの開始を示している．13 行目の nodeName は c0a22100-master で，ジョブが必ずこのノードで実行されるように指定している．14 行目の containers は Pod 内で実行されるコンテナのリストを定義するフィールドである．15 行目の- name は postgresql-backup でコンテナの名前を指定している．16 行目の image は c0a22100/postgresql-backup:latest で使用する Docker イメージを指定している．17 行目の command は ['bash', '/backup.sh'] で，コンテナ起動時に実行するコマンドを指定している．18 行目から 29 行目は，env は環境変数を定義するフィールドであり，DB_HOST・DB_USER・DB_PASS・DB_NAME が指定されている．DB_PASS は Secret（postgresql-backup-secret）から参照されるように設定されている．30 行目の volumeMounts はコンテナにマウントするボリュームを定義するフィールドである．31 行目の- name は backup-volume で，マウント対象のボリューム名を指定している．32 行目の mountPath は/backup で，コンテナ内のマウント先ディレクトリを指定している．33 行目の volumes は Pod で利用するボリュームのリストを定義するフィールドである．34 行目の- name は backup-volume で，ボリュームの名前を指定している．35 行目の nfs は NFS サーバーを利用してボリュームをマウントする設定を示している．36 行目の server は 192.168.100.5 で，NFS サーバーの IP アドレスを指定している．37 行目の path は/volume1/c0a22100-postgresql-backup で，NFS サーバー上のディレクトリパスを指定している．38 行目の restartPolicy は OnFailure でジョブが失敗した場合にのみ再起動するポリシーを指定している．

プログラム 9は，記述ミスのある YAML ファイルの kubectl describe の結果である．kubectl apply コマンドで記述ミスのある YAML ファイルを Kubernetes クラスタに適用する．適用後の Pod に対して kubectl describe コマンドを実行した結果の一部がプログラム 9である．プログラム 9では，Events の Type における Warning は，異常や問題が発生していることを表している．Events の Reason における BackOff は，コンテナの起動に失敗し続けていて，再起動を一定時間待機していることを表している．Events の Age における 4s (x4 over 36s) は，過去 36 秒間に同じイベントが 4 回発生したことを表している．

プログラム 8 記述ミスのある YAML ファイル

```
1  apiVersion: batch/v1
2  kind: CronJob
3  metadata:
4   name: postgresql-backup
5   namespace: ssh-monitoring
6  spec:
7   # schedule: "00 19 * * *"
8   schedule: "*/1 * * * *"
9   jobTemplate:
10    spec:
11      template:
12        spec:
13          nodeName: c0a22100-master
14          containers:
15          - name: postgresql-backup
16            image: c0a22100/postgresql-backup:
                latest
17            command: ['bash', '/backup.sh']
18            env:
19            - name: DB_HOST
20              value: c0a22100-main:30088
21            - name: DB_USER
22              value: postgres
23            - name: DB_PASS
24              valueFrom:
25                secretKeyRef:
26                  name: postgresql-backup-secret
27                  key: db_password
28            - name: DB_NAME
29              value: ssh_monitoring
30            volumeMounts:
31            - name: backup-volume
32              mountPath: /backup
33          volumes:
34          - name: backup-volume
35            nfs:
36              server: 192.168.100.5 # Tapioca
37              path: "/volume1/c0a22100-
                postgresql-backup"
38          restartPolicy: OnFailure
```

プログラム 9 記述ミスのある YAML ファイルの kubectl describe の結果

```
1  Events:
2    Type Reason Age From Message
3    ---- ------ ---- ---- -------
4    Warning BackOff 4s (x4 over 36s) kubelet
     Back-off restarting failed container
     postgresql-backup in pod postgresql-backup
     -29314228-nhv4z_ssh-monitoring(2e83acc6-f837
     -4b89-9c09-c9b40a55261a)
```

Events の From における kubelet は，ノード上の kubelet が「コンテナの起動失敗」を検知していることを表している．Events の Message における Back-off restarting

failed container postgresql-backup in pod postgresql-backup-29314228-nhv4z_ssh-monitoring(2e83acc6-f837-4b89-9c09-c9b40a55261a) は，Pod 内のコンテナが起動に失敗し続けていて，再起動をバックオフしていることを表している．

プログラム 10は，記述ミスのある YAML ファイルの kubectl describe の結果における Events の Message である．プログラム 10では，Events の Message にコンテナ起動に失敗した時のメッセージが出力されている．Message を空白区切りで単語列に分割したものと各単語がルールに定義された keyword_list を比較することで，ルールに含まれるエラーか判定している．

プログラム 11は，コンテナの起動に繰り返し失敗するケースのルールである．プログラム 11では，Events の Message に含まれる単語列と keyword_list の共通部分が70%以上存在する場合に，ルール「Back-off restarting failed container」にマッチしたと判定する．ルールのキーワードリストは，[”Back-off”, ”restarting”, ”failed”, ”container”, ”dns-backup”, ”in”, ”pod”, ”dns-backup-29422560-f6zfr_default(f09e6188-0a07-47d1-b6fc-d4a78309eeda)”] である．kubectl describe コマンドの結果は，[”Back-off”, ”restarting”, ”failed”, ”container”, ”postgresql-backup”, ”in”, ”pod”, ”postgresql-backup-29314228-nhv4z_ssh-monitoring(2e83acc6-f837-4b89-9c09-c9b40a55261a)”] である．ルールのキーワードリストに含まれる単語の総数は 8 個である．ルールのキーワードリストと kubectl describe コマンドの結果で一致する単語は，Back-off, restarting，failed，container，in，pod で 6 個である．ルールのキーワードリストと kubectl describe コマンドの結果で一致しない単語は，コンテナ名や Pod 名は postgresql-backup-29314228-nhv4z_ssh-monitoring と dns-backup-29422560-f6zfr_default の 2 個である．そのため，ルールのキーワードリストと kubectl describe コマンドの結果の共通部分は 6/8 = 0.75 により 75%であり，一致したと判定する．

プログラム 12は提案ソフトウェアの出力例を表している．プログラム 12では，1 行目から 5 行目の記述ミスのある YAML ファイルの一部は，記述した YAML ファイルの中でエラーの原因箇所を起点として，前後 2 行を出力している．7 行目の説明は，記述ミスのある YAML ファイルの何行目がエラーの原因箇所であるのか，1 行目から 4 行目の出力を活用してメッセージを出力している．出力している 1 行目から 4 行目の記述ミスのある YAML ファイルの一部における 17 行目は，記述したコンテナ内のパス指定を間違えている．YAML ファイルの 17 行目は，command: [’bash’, ’/backup.sh’] と記述すべきではない．記述ミスのある YAML ファイルの一部における 17 行目の正しいコンテナ内のパス指定は，command: [’/usr/local/bin/backup.sh’] である．

プログラム 10 記述ミスのある YAML ファイルの kubectl describe の結果における Events の Message

```
Events:
  Type Reason Age From Message
  ---- ------ ---- ---- -------
  Warning BackOff 4s (x4 over 36s) kubelet
  Back-off restarting failed container
  postgresql-backup in pod postgresql-backup
  -29314228-nhv4z_ssh-monitoring(2e83acc6-f837
  -4b89-9c09-c9b40a55261a)
```

プログラム 11 コンテナの起動に繰り返し失敗するケースのルール

```
{
  "name": "Back-off restarting failed
  container",
  "keyword_list": ["Back-off","restarting","
  failed","container","dns-backup","in","pod
  ","dns-backup-29422560-f6zfr_default(
  f09e6188-0a07-47d1-b6fc-d4a78309eeda)"],
  "parameter": "spec.jobTemplate.spec.template
  .spec.containers.command"
}
```

プログラム 12 提案ソフトウェアの出力例

```
15 - name: postgresql-backup
16 image: c0a22100/postgresql-backup:latest
17 command: ['bash', '/backup.sh']
18 env:
19 - name: DB_HOST

YAML ファイルの 17 行目に記述されている command: ['bash', '/backup.sh']が原因である可能性があります.
```

ユースケース・シナリオ

図 5は，ユースケース・シナリオを示している．図 5では，佐藤さん，作業するサーバ，YAML ファイル，シェルスクリプト，kubectl apply -f prometheus-cronjob.yaml，入力するルールが保存されたファイル，入力する YAML ファイル，入力する kubectl describe コマンドに結果，提案ソフトウェア，エラーの原因箇所で構成されている．佐藤さんは，CDSL の佐藤健斗さんを表す．作業するサーバは，佐藤さんが開発作業をする Kubernetes クラスタ内のサーバを表す．YAML ファイルは，開発作業で作成された YAML ファイルを表す．kubectl apply -f prometheus-cronjob.yaml は，開発作業で作成された YAML ファイルを用いて Pod を起動させるために実行するコマンドである．入力するルールが保存されたファイルは，複数のエラーパターンを用いて作成したファイルである．入力する kubectl describe コマンドの結果は，記述ミスのある YAML ファイルを用いて

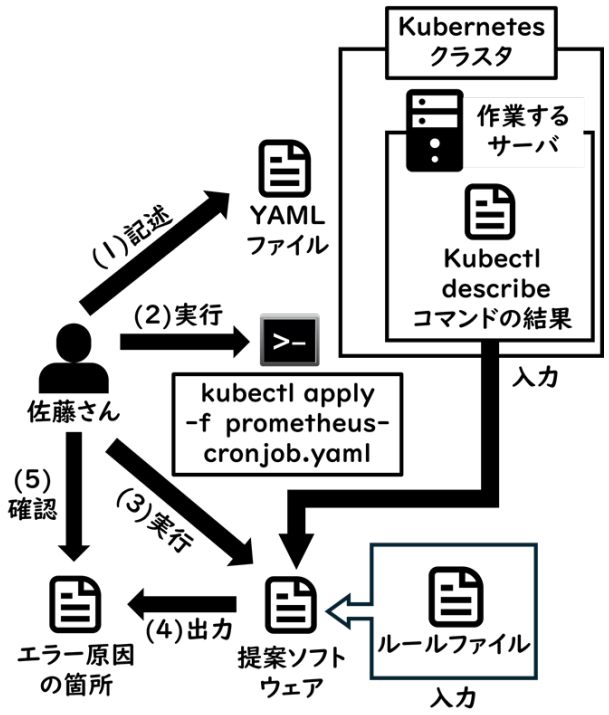

図 5　ユースケース・シナリオ

起動した Pod に対して実行するコマンドを表す．提案ソフトウェアは，開発作業を効率化するためのソフトウェアを表す．エラーの原因箇所は，提案ソフトウェアを用いて出力されたメッセージを表す．(1) では，佐藤さんが YAML ファイルとシェルスクリプトに記述している．(2) では，佐藤さんが開発作業で作成された YAML ファイルを kubectl apply -f prometheus-cronjob.yaml で実行する．(3) では，佐藤さんが入力として (1) の開発作業で作成された YAML ファイルと kubectl describe コマンドの結果とルールが保存されたファイルの 3 つのファイルを指定して提案ソフトウェアを実行する．(4) では，提案ソフトウェアからエラーの原因箇所が出力される．(5) では，佐藤さんがエラーの原因箇所を確認する．出力として提案ソフトウエアからエラーの原因箇所をメッセージで送信する．

4. 実装

図 6は，実装したシステムの構成を示している．図 6では，プロンプト，生成 AI，miss_add.yaml，Kubernetes クラスタ，kubectl describe コマンドの結果，rule_create.py，rule.txt，matching.py，miss.yaml，エラーの原因箇所で構成されている．プロンプトは，生成 AI に何をしてもらうのか指示をする文章である．生成 AI は，大規模言語モデルをもとにして出現確率を計算することでテキストを生成する技術である．開発作業において発生するエラーに対して，最大限に対策するため，生成 AI を使用してルールを生成する．miss_add.yaml は，プロンプトの指示を受けて生成 AI が作成した YAML ファイルである．Kubernetes

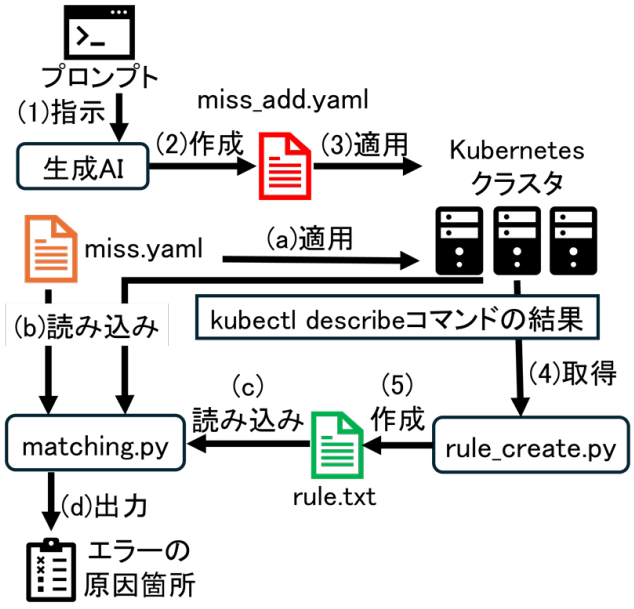

図 6　実装したシステムの構成

クラスタは，3 つのノードで構成される．kubectl describe コマンドの結果は，記述ミスを追加した YAML ファイルと記述ミスのある YAML ファイルを適用した後にコマンドを実行した際の結果である．rule_create.py は，kubectl describe コマンドの結果からルールを作成するためのプログラムである．ルールはルール名やキーワードリストが含まれている．matching.py は，ルールと記述ミスのある YAML ファイルと kubectl describe の結果を比較して一致するか確認するプログラムである．miss.yaml は，開発作業で作成された記述ミスを含む YAML ファイルである．エラーの原因箇所は，ルールと記述ミスのある YAML ファイル，kubectl describe コマンドの結果を比較した後に出力される．エラーの原因箇所には YAML ファイルの行番号と行の内容が含まれる．

rule_create.py の動作の順序を以下に示す．

(1) 生成 AI にプロンプトで指示を送る．
(2) 生成 AI で miss_add.yaml を作成する．
(3) miss_add.yaml を Kubernetes クラスタで適用する．
(4) rule_create.py によって kubectl describe コマンドの結果を取得する．
(5) rule_create.py によって rule.txt を作成する．

matching.py の動作の順序を以下に示す．

(a) matching.py によって記述ミスのある YAML ファイルを Kubernetes クラスタで適用して，kubectl describe コマンドの結果を取得する．
(b) matching.py によって miss.yaml を読み込む．
(c) matching.py によって rule.txt を読み込む．

プログラム 13に実装したシステムの構成のルールを示す．プログラム 13では，JSON 形式で 2 行目は Key を name とし

て値を Back-off restarting failed container とする．Back-off restarting failed container は，kubectl describe コマンドの結果における Events の Message の初めから 4 単語までを抽出したものである．3 行目は Key を keyword_list として値を [”Back-off”,”restarting”,”failed”,”container”,”dns-backup”,”in”,”pod”,”dns-backup-29422560-f6zfr_default (f09e6188-0a07-47d1-b6fc-d4a78309eeda)”] とする．この keyword_list はマッチングプログラムに使用するキーワードリストを表す．4 行目は Key を parameter として 値 を”spec.jobTemplate.spec.template.spec.containers .command”とする．parameter は，マッチングプログラムで記述ミスのある YAML ファイルにおけるエラーの原因箇所を探すために設定している．

プログラム 13　実装したシステムの構成のルール

```
1 {
2 "name": "Back-off restarting failed container
  ",
3 "keyword_list": ["Back-off","restarting","
  failed","container","dns-backup","in","pod","
  dns-backup-29422560-f6zfr_default(f09e6188-0
  a07-47d1-b6fc-d4a78309eeda)"],
4 "parameter": "spec.jobTemplate.spec.template.
  spec.containers.command"
5 }
```

プログラム 14　提案ソフトウェアの出力例

```
1 15 - name: postgresql-backup
2 16 image: c0a22100/postgresql-backup:latest
3 17 command: ['bash', '/backup.sh']
4 18 env:
5 19 - name: DB_HOST
6
7 YAML ファイルの 17 行目に記述されている command:
  ['bash', '/backup.sh']が原因である可能性があります.
```

プログラム 14は，提案ソフトウェアの出力例を示している．プログラム 14では，1 行目から 5 行目の記述ミスのある YAML ファイルの一部は，記述した YAML ファイルの中でエラーの原因箇所を起点として，前後 2 行を出力している．7 行目の説明は，記述ミスのある YAML ファイルの何行目がエラーの原因箇所であるのか，1 行目から 4 行目の出力を活用してメッセージを出力している．出力している 1 行目から 5 行目の記述ミスのある YAML ファイルの一部における 17 行目は，記述したコンテナ内のパス指定を間違えている．YAML ファイルの 17 行目は，command: ['bash', '/backup.sh'] と記述すべきではない．記述ミスのある YAML ファイルの一部における 17 行目の正しいコンテナ内のパス指定は，command: ['/usr/local/bin/backup.sh'] である．

5. 評価実験

評価実験では，評価用の YAML ファイル（評価ファイル）のうち，提案ソフトウェアが記述ミスを検出した割合を検出率で評価する．検出率は，評価ファイルの数を N として，提案ソフトウェアが検出した数を P として，P/N で求める．

実験環境

仮想化環境にインストールした Ubuntu の上にマスターノードとワーカーノードを作成した．VMware ESXi のホスト名である Lotus では，マスターとワーカーとして 2 台の仮想マシンを作成した．そして，2 台の仮想マシンには，以下の項目で設定した．

- ゲスト OS：Ubuntu 24.04.2 LTS
- vCPU：2core
- メモリ：8GB
- ディスク容量：30GB

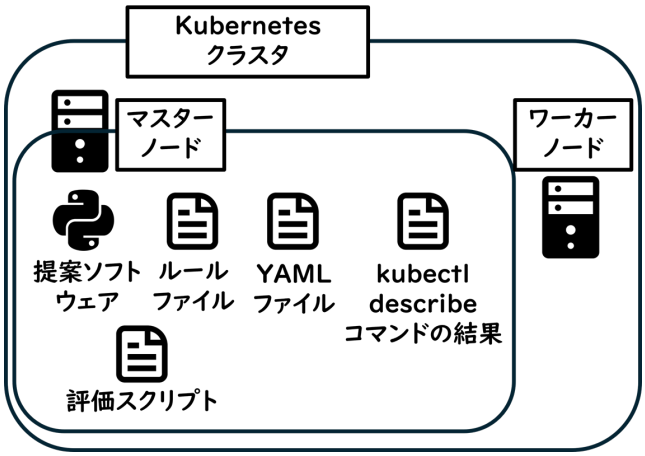

図 7　実験環境

図 7は，実験環境を示している．図 7では，提案ソフトウェア，ルールが保存されたファイル，YAML ファイル，kubectl describe コマンドの結果，評価スクリプトで構成されている．提案ソフトウェアは，記述ミスのある YAML ファイルと kubectl describe コマンドの結果を用いてエラーの原因箇所を出力するソフトウェアである．ルールが保存されたファイルは，生成 AI にプロンプトで指示を送って作成されたファイルである．YAML ファイルは，記述ミスのある YAML ファイルである．kubectl describe コマンドの結果は，YAML ファイルを kubectl apply コマンドで Pod を起動しようとした際に，Pod の状態確認をするためのコマンドである．評価スクリプトは，記述ミスのある YAML ファイルと kubectl describe コマンドの結果の二つで評価実験をするためのスクリプトである．Kubernetes クラスタを構築するために以下の手順を行う．

(1) K3s のインストールスクリプトを実行
(2) クラスタが正常に動作しているか確認
(3) ワーカーノードで環境変数を指定してインストール
(1) では，マスターノードでプログラム 15を実行して K3s をインストールする．(2) では，マスターノードでプログラム 16を実行してクラスタが正常に動作しているか確認する．(3) では，ワーカーノードでプログラム 17を実行して複数のノード構成にする．server-ip はマスターノードの IP アドレスを記述する．token はマスターノードで cat /var/lib/rancher/k3s/server/node-token を実行して表示されたトークンを記述する．

プログラム 15 K3s のインストールスクリプトを実行

```
curl -sfL https://get.k3s.io | sh -s -
```

プログラム 16 クラスタが正常に動作しているか確認

```
kubectl get nodes
```

プログラム 17 ワーカーノードで環境変数を指定してインストール

```
curl -sfL https://get.k3s.io | K3S\_URL=https
://server-ip:6443 K3S\_TOKEN=token sh -
```

データセット

表 1は，評価用のデータセットを示している．表 1は，評価ファイルと記述ミスのある箇所または変更した箇所の 2 つの項目が含まれている．20250912-tukimori-1 は, 2025 年 9 月 12 日に開発作業をした月森さんが記述ミスをした 1 つ目の YAML ファイルを表し，記述ミスのある箇所は restartPolicy のインデントが違うことを示す．20250912-tukimori-2 は，2025 年 9 月 12 日に開発作業をした月森さんが記述ミスをした 2 つ目の YAML ファイルを表し，記述ミスのある箇所は volumes の name が誤って記述されていることを示す．20250919-tukimori-1 は，2025 年 9 月 19 日に開発作業をした月森さんが修正した 1 つ目の YAML ファイルを表し，変更した箇所として nodeName の追加と nfs の設定を行ったことを示す．20250919-tukimori-2 は，2025 年 9 月 19 日に開発作業をした月森さんが修正した 2 つ目の YAML ファイルを表し，変更した箇所として containers の env の DB_HOST で value のポート番号を変更したことを示す．20250919-tukimori-3 は，2025 年 9 月 19 日に開発作業をした月森さんが修正した 3 つ目の YAML ファイルを表し，変更した箇所として containers の env の DB_HOST で value のポート番号を変更したことを示す．20250922-satou-1 は，2025 年 9 月 22 日に開発作業をした佐藤さんが修正した 1 つ目の YAML ファイルを表し，変更した箇所として cronjob のスケジュールを変更したことを示す．20250924-tukimori-1 は，2025 年 9 月 24 日に開発作業をした月森さんが修正した 1 つ目の YAML ファイルを表し，変更した箇所として nodeName を追加したことを示す．20250924-tukimori-2 は，2025 年 9 月 24 日に開発作業をした月森さんが修正した 2 つ目の YAML ファイルを表し，変更した箇所として nodeName を追加したことを示す．20250924-tukimori-3 は，2025 年 9 月 24 日に開発作業をした月森さんが修正した 3 つ目の YAML ファイルを表し，変更した箇所としてスケジュールと DB_PASS を変更したことを示す．20250925-tukimori-1 は，2025 年 9 月 25 日に開発作業をした月森さんが修正した 1 つ目の YAML ファイルを表し，変更した箇所として nodeName を追加したことを示す．20250925-tukimori-2 は，2025 年 9 月 25 日に開発作業をした月森さんが修正した 2 つ目の YAML ファイルを表し，変更した箇所として containers の command を変更したことを示す．20250925-tukimori-3 は，2025 年 9 月 25 日に開発作業をした月森さんが修正した 3 つ目の YAML ファイルを表し，変更した箇所として containers の env の DB_HOST において value のホスト名を変更したことを示す．

表 1 評価用のデータセット

評価ファイル	記述ミスのある箇所または変更した箇所
20250912-tukimori-1	restartPolicy のインデントが違う
20250912-tukimori-2	volumes の name が違う
20250919-tukimori-1	nodeName を追加している& nfs の設定をしている
20250919-tukimori-2	containers の env の DB_HOST で value のポート番号を変更
20250919-tukimori-3	containers の env の DB_HOST で value のポート番号を変更
20250922-satou-1	cronjob のスケジュールを変更
20250924-tukimori-1	nodeName の追加
20250924-tukimori-2	nodeName の追加
20250924-tukimori-3	スケジュールと DB_PASS の変更
20250925-tukimori-1	nodeName の追加
20250925-tukimori-2	containers の command を変更
20250925-tukimori-3	containers の env の DB_HOST で value のホスト名を変更

実験結果と分析

実験結果は，12 個の評価ファイルのうち，5 個の評価ファイルが 70%以上の一致率であった．一致率は，kubectl describe コマンドの Events における Message とルールのキーワードリストを比較して算出した割合である．

表 2は，評価ファイルの一致率を示している．一致率の数値がある場合は一致している割合を示し，- はデータがないことを示す．データがない理由は，kubectl describe コマンドの結果における Events の Message を含む行の抽出に失敗しているからである．20250912-tukimori-1 は, 2025 年 9 月 12 日に開発作業をした月森さんが記述ミスをした 1 つ目の YAML ファイルを表し，データがないことを示

表 2　評価ファイルの一致率

評価ファイル	一致率
20250912-tukimori-1.yaml	-
20250912-tukimori-2.yaml	-
20250919-tukimori-1.yaml	0.12
20250919-tukimori-2.yaml	0.88
20250919-tukimori-3.yaml	0.88
20250922-satou-1.yaml	-
20250924-tukimori-1.yaml	0.12
20250924-tukimori-2.yaml	0.12
20250924-tukimori-3.yaml	0.88
20250925-tukimori-1.yaml	0.12
20250925-tukimori-2.yaml	0.75
20250925-tukimori-3.yaml	0.75

す．20250912-tukimori-2 は，2025 年 9 月 12 日に開発作業をした月森さんが記述ミスをした 2 つ目の YAML ファイルを表し，データがないことを示す．20250919-tukimori-1 は，2025 年 9 月 19 日に開発作業をした月森さんが修正した 1 つ目の YAML ファイルを表し，12%の一致率を示す．20250919-tukimori-2 は，2025 年 9 月 19 日に開発作業をした月森さんが修正した 2 つ目の YAML ファイルを表し，88%の一致率を示す．20250919-tukimori-3 は，2025 年 9 月 19 日に開発作業をした月森さんが修正した 3 つ目の YAML ファイルを表し，88%の一致率を示す．20250922-satou-1 は，2025 年 9 月 22 日に開発作業をした佐藤さんが修正した 1 つ目の YAML ファイルを表し，データがないことを示す．20250924-tukimori-1 は，2025 年 9 月 24 日に開発作業をした月森さんが修正した 1 つ目の YAML ファイルを表し，12%の一致率を示す．20250924-tukimori-2 は，2025 年 9 月 24 日に開発作業をした月森さんが修正した 2 つ目の YAML ファイルを表し，12%に一致率を示す．20250924-tukimori-3 は，2025 年 9 月 24 日に開発作業をした月森さんが修正した 3 つ目の YAML ファイルを表し，88%の一致率を示す．20250925-tukimori-1 は，2025 年 9 月 25 日に開発作業をした月森さんが修正した 1 つ目の YAML ファイルを表し，12%の一致率を示す．20250925-tukimori-2 は，2025 年 9 月 25 日に開発作業をした月森さんが修正した 2 つ目の YAML ファイルを表し，75%の一致率を示す．20250925-tukimori-3 は，2025 年 9 月 25 日に開発作業をした月森さんが修正した 3 つ目の YAML ファイルを表し，75%の一致率を示す．

プログラム 18は，20250919-tukimori-1.yaml のキーワードリストを示している．キーワードリストは，起動している Pod に対して kubectl describe コマンドを実行して，出力された結果の Events における Message をカンマ区切りで保存したものである．このキーワードリストは NFS のマウントに失敗したことをあらわす．

プログラム 19は，ルールのキーワードリストを示している．このキーワードリストは Pod の再起動に失敗したことをあらわす．キーワードリストに含まれる単語は，Back-off, restarting, failed, container, dns-backup, in, pod, dns-backup-29426065-m6lrn_default(0b41c6ce-5a8b-415c-bd1b-5cec9ed6cc8e) の 8 つである．プログラム 18には，8 単語のうち，”failed”が含まれている．したがって，一致率は 1 / 8 = 0.125 であるので，約 12%を示している．

プログラム 18　20250919-tukimori-1.yaml のキーワードリスト

```
1 ["MountVolume.SetUp","failed","for","volume
  ","\"backup-volume\"",":","mount","failed:","
  exit","status","32","Mounting","command:","
  mount","Mounting","arguments:","-t","nfs
  ","192.168.100.5:/volume1/dns-backup","/var/
  lib/kubelet/pods/7ff23751-6c90-4bff-9629-542
  ad42ec5e2/volumes/kubernetes.io~nfs/backup-
  volume","Output:","mount:","/var/lib/kubelet/
  pods/7ff23751-6c90-4bff-9629-542ad42ec5e2/
  volumes/kubernetes.io~nfs/backup-volume:","bad
  ","option;","for","several","filesystems","(e.
  g.","nfs,","cifs)","you","might","need","a","/
  sbin/mount.<type>","helper","program."]
```

プログラム 19　ルールのキーワードリスト

```
1 ["Back-off","restarting","failed","container
  ","dns-backup","in","pod","dns-backup
  -29426065-m6lrn_default(0b41c6ce-5a8b-415c-
  bd1b-5cec9ed6cc8e)"]
```

表 3　評価実験の結果

評価ファイル	検出結果
20250912-tukimori-1	×
20250912-tukimori-2	×
20250919-tukimori-1	×
20250919-tukimori-2	○
20250919-tukimori-3	○
20250922-satou-1	×
20250924-tukimori-1	×
20250924-tukimori-2	×
20250924-tukimori-3	○
20250925-tukimori-1	×
20250925-tukimori-2	○
20250925-tukimori-3	○

表 3は，評価実験の結果を示している．表 3は，評価ファイルと検出結果の 2 つの項目が含まれている．検出結果の○は検出されていることを示し，検出結果の×は検出されていないことを示す．20250912-tukimori-1 は, 2025 年 9 月 12 日に開発作業をした月森さんが記述ミスをした 1 つ目の YAML ファイルを表し，検出されていないことを示す．20250912-tukimori-2 は，2025 年 9 月 12 日に開発作業をした月森さんが記述ミスをした 2 つ目の YAML ファイルを表し，検出されていないことを示す．20250919-tukimori-1

は，2025 年 9 月 19 日に開発作業をした月森さんが修正した 1 つ目の YAML ファイルを表し，検出されていないことを示す．20250919-tukimori-2 は，2025 年 9 月 19 日に開発作業をした月森さんが修正した 2 つ目の YAML ファイルを表し，検出されていることを示す．20250919-tukimori-3 は，2025 年 9 月 19 日に開発作業をした月森さんが修正した 3 つ目の YAML ファイルを表し，検出されていることを示す．20250922-satou-1 は，2025 年 9 月 22 日に開発作業をした佐藤さんが修正した 1 つ目の YAML ファイルを表し，検出されていないことを示す．20250924-tukimori-1 は，2025 年 9 月 24 日に開発作業をした月森さんが修正した 1 つ目の YAML ファイルを表し，検出されていないことを示す．20250924-tukimori-2 は，2025 年 9 月 24 日に開発作業をした月森さんが修正した 2 つ目の YAML ファイルを表し，検出されていないことを示す．20250924-tukimori-3 は，2025 年 9 月 24 日に開発作業をした月森さんが修正した 3 つ目の YAML ファイルを表し，検出されていることを示す．20250925-tukimori-1 は，2025 年 9 月 25 日に開発作業をした月森さんが修正した 1 つ目の YAML ファイルを表し，検出されていないことを示す．20250925-tukimori-2 は，2025 年 9 月 25 日に開発作業をした月森さんが修正した 2 つ目の YAML ファイルを表し，検出されていることを示す．20250925-tukimori-3 は，2025 年 9 月 25 日に開発作業をした月森さんが修正した 3 つ目の YAML ファイルを表し，検出されていることを示す．評価実験の結果から検出率を算出する．検出率は，P の値は 5 で，N の値は 12 であるため，$5/12 \approx 0.417$ である．したがって，検出率は約 41.7%である．

6. 議論

提案方式では，生成 AI を用いてルールを作成し，記述ミスのある YAML ファイルと kubectl describe コマンドの結果を比較してエラーの原因箇所を特定している．評価実験の結果では 12 個の記述ミスのある YAML ファイルに対して，エラーの原因箇所を全く検出できなかった．原因は，ルールが保存されたファイルに含まれる正規表現で成形した値が kubectl describe コマンドの結果と一致しなかった点にある．この問題に対して，ルールに含まれる正規表現で成形した値を使用せずキーワードリストを使用する．キーワードリストには，kubectl describe コマンドの結果の Events に含まれる Message から取り出した単語がカンマ区切りで含まれる．また，生成 AI に指示を送るプロンプトに含まれるルールの定義および出力例のルールを変更する．具体的には，ルールに含まれるパターンをキーワードリストとして設定し，出力例のルールでも同様にパターンの項目をキーワードリストとして設定する．

提案方式では，生成 AI を使いルールの作成を行った．生成 AI の出力の正確性の検証が不足している．提案方式の場合，ハルシネーションが起きると生成 AI の出力やルールが保存されたファイルに誤りが含まれる．誤りが含まれるルールが保存されたファイルを使用すると検出率が低下する．この問題に対して，生成 AI でキーワードリストを作成する方法の代わりに，単語を kubectl describe コマンドの結果における Events の Message から取り出すプログラムを作成し使用する．まず，生成 AI を用いて記述ミスのある YAML ファイルを生成する．次に，生成された YAML ファイルをテスト用の Kubernetes クラスタに対して kubectl apply コマンドで適用し，Pod を起動させる．その後，kubectl describe コマンドを実行して Event，Reason，Message を含むエラーの詳細を取得する．さらに，Message に含まれる単語をカンマ区切りで分けてキーワードリストを作成する．その作成したキーワードリストをルールに保存する．

提案方式では，生成 AI を用いてルールを作成して，記述ミスのある YAML ファイルと kubectl describe コマンドの結果を比較してエラーの原因箇所を特定している．提案方式で生成したルールが実際に起きるエラーをどの程度網羅しているか検出率に影響する．収集した記述ミスのない YAML ファイルの件数が増えると，ルールの数が増えて検出率があがる．記述ミスのない YAML ファイルの種類が少ない場合には，生成されるルールが特定のエラーに偏り，未知のエラーを検出できないという問題がある．この問題に対して，インターネット上に公開されている記述ミスのない YAML ファイルを収集するだけではなく，「正解のスキーマと比較してどこが異なるのか」のような観点を取り入れることで，エラーの特徴を体系化できるアプローチが有効である．Kubernetes 公式ドキュメントには，CronJob，Deployment，Pod のリソースに正しい構造を定義したスキーマ（OpenAPI Schema）が存在しており，そのスキーマを基準とすることで YAML の構造的誤り，必須項目の欠落，型の不一致を網羅的に検出できる．

7. おわりに

課題では，バックアップを構築する過程において Pod の状況を確認してから kubectl apply コマンドを実行するまでに時間がかかっている．基礎実験では，佐藤さんの開発作業において，Pod の状況確認に費やした時間は合計 900 秒であり，関連作業全体の 50%を占めていた．このことから，Pod の状況確認が作業時間の大きな割合を占めていることが分かる．提案方式では，エラーの原因特定にかかる時間を短縮するために，ルールを用いたパターンマッチングにより YAML ファイルのエラーの原因箇所を特定する．評価実験では，12 個の開発作業で作成された記述ミスを含む YAML ファイルを対象に，提案ソフトウェアが検出した割合を検出率で評価した．検出率は，12 個のうち 5 個検出されたため，約 41.7%であった．

参考文献

[1] Carrión, C.: Kubernetes Scheduling: Taxonomy, Ongoing Issues and Challenges, *ACM Comput. Surv.*, Vol. 55, No. 7 (online), DOI: 10.1145/3539606 (2022).

[2] Menouer, T.: KCSS: Kubernetes container scheduling strategy, *The Journal of Supercomputing*, Vol. 77, pp. 4267–4293 (online), DOI: 10.1007/s11227-020-03427-3 (2021).

[3] Chiba, T., Nakazawa, R., Horii, H., Suneja, S. and Seelam, S.: ConfAdvisor: A Performance-centric Configuration Tuning Framework for Containers on Kubernetes, *2019 IEEE International Conference on Cloud Engineering (IC2E)*, pp. 168–178 (online), DOI: 10.1109/IC2E.2019.00031 (2019).

[4] Predoaia, I., Kolovos, D., Garcia-Dominguez, A., Lenk, M., Ebel, W. and Burkl, J.: Towards Processing YAML Documents with Model Management Languages, *Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems*, MODELS Companion '24, New York, NY, USA, Association for Computing Machinery, p. 970–979 (online), DOI: 10.1145/3652620.3688219 (2024).

[5] Barletta, M., Cinque, M., Di Martino, C., Kalbarczyk, Z. T. and Iyer, R. K.: Mutiny! How Does Kubernetes Fail, and What Can We Do About It?, *2024 54th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 1–14 (online), DOI: 10.1109/DSN58291.2024.00016 (2024).

[6] Ghosh, S., Shetty, M., Bansal, C. and Nath, S.: How to fight production incidents? an empirical study on a large-scale cloud service, *Proceedings of the 13th Symposium on Cloud Computing*, SoCC '22, New York, NY, USA, Association for Computing Machinery, p. 126–141 (online), DOI: 10.1145/3542929.3563482 (2022).

[7] Zhao, N., Chen, J., Yu, Z., Wang, H., Li, J., Qiu, B., Xu, H., Zhang, W., Sui, K. and Pei, D.: Identifying bad software changes via multimodal anomaly detection for online service systems, *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2021, New York, NY, USA, Association for Computing Machinery, p. 527–539 (online), DOI: 10.1145/3468264.3468543 (2021).

[8] 田中美帆，小山智之，串田高幸：YAML ファイルを比較して生成したルールによる Kubernetes Pod のエラー原因の提示，Technical Report CDSL-TR-226, CDSL (2025).

[9] Predoaia, I., Kolovos, D., Garcia-Dominguez, A., Lenk, M., Ebel, W. and Burkl, J.: Towards Processing YAML Documents with Model Management Languages, *Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems*, MODELS Companion '24, New York, NY, USA, Association for Computing Machinery, p. 970–979 (online), DOI: 10.1145/3652620.3688219 (2024).

[10] Mahajan, V. B. and Mane, S. B.: Detection, Analysis and Countermeasures for Container based Misconfiguration using Docker and Kubernetes, *2022 International Conference on Computing, Communication, Security and Intelligent Systems (IC3SIS)*, pp. 1–6 (online), DOI: 10.1109/IC3SIS54991.2022.9885293 (2022).

[11] Ranković, T., Šiljić, F., Tomić, J., Sladić, G. and Simić, M.: Misconfiguration Prevention and Error Cause Detection for Distributed-Cloud Applications, *2024 IEEE 22nd Jubilee International Symposium on Intelligent Systems and Informatics (SISY)*, pp. 000297–000302 (online), DOI: 10.1109/SISY62279.2024.10737513 (2024).