Научная статья
УДК 004.451.26
https://doi.org/10.24143/2072-9502-2024-4-35-43
EDN WHFYOS

Использование алгоритма множественной линейной регрессии и паттернов проектирования для составления файлов конфигурации кластера Kubernetes

Д. С. Фомин, А. В. Бальзамов, А. В. Савкина, С. А. Федосин, В. В. Никулин ✉

Национальный исследовательский Мордовский государственный университет им. Н. П. Огарёва,
Саранск, Россия, nikulinvv@mail.ru ✉

Аннотация. Рассматривается проблема конфигурации кластеров Kubernetes. Так как настройки кластера производятся с помощью конфигурационных файлов YAML, содержащих в себе большое количество параметров, ссылок на репозитории (открытые и закрытые) и внешние источники данных, то довольно просто допустить ошибку, которая может привести к существенным издержкам в будущем. Когда все необходимые данные для файла корректно подготовлены, необходимо их правильно скомпоновать в соответствии с синтаксисом разметки YAML. Цель работы – поиск оптимального метода автоматизации построения конфигурационного файла Kubernetes на основе статистических данных. Проведен анализ проблемы конфигурации Kubernetes на основе YAML-файлов и проблемы интерпретации статистических данных в структурированный файл YAML и предложены конкретные методы и подходы по решению указанных проблем. Приводится измененный алгоритм множественной линейной регрессии для работы с собранными статистическими данными, результат выходных данных алгоритма и блок-схема паттерна, адаптированного для построения YAML-файлов. Предложенные подходы позволяют использовать дополнительные инструменты для работы с тестовыми и рабочими кластерами Kubernetes, что позволяет снизить сложность взаимодействия разработчиков с ними и повысить скорость развертывания и масштабируемость. Кроме того, описанные методы позволяют упростить администрирование крупных сетей и автоматизировать процесс создания конфигурационных YAML-фалов для популярных шаблонов программного обеспечения.

Ключевые слова: алгоритм, файл, конфигурация, библиотека классов, паттерн

Для цитирования: *Фомин Д. С., Бальзамов А. В., Савкина А. В., Федосин С. А., Никулин В. В.* Использование алгоритма множественной линейной регрессии и паттернов проектирования для составления файлов конфигурации кластера Kubernetes // Вестник Астраханского государственного технического университета. Серия: Управление, вычислительная техника и информатика. 2024. № 4. С. 35–43. https://doi.org/10.24143/2072-9502-2024-4-35-43. EDN WHFYOS.

Original article

Using multiple linear regression algorithm and design patterns to complete Kubernetes cluster configuration files

D. S. Fomin, A. V. Balzamov, A. V. Savkina, S. A. Fedosin, V. V. Nikulin ✉

National Research Ogarev Mordovia State University,
Saransk, Russia, nikulinvv@mail.ru ✉

Abstract. The problem of configuration of Kubernetes clusters is considered. Since cluster settings are made using YAML configuration files containing a large number of parameters, links to repositories (open and closed) and external data sources, it is quite easy to make a mistake that will incur significant costs in the future. When all the necessary data for the file is correctly prepared, it is necessary to arrange them correctly in accordance with the syntax of the YAML markup. The purpose of the work is to search for the optimal method of automati-zation of building a Kubernetes configuration file based on statistical data.The analysis of the Kubernetes configuration problem based on YAML files and the problem of interpreting statistical data into a structured YAML file is carried out and specific methods and approaches to solving these problems are proposed. The modified algorithm of multiple linear regression for working with the collected

statistical data, the result of the output data of the algorithm and the flowchart of the pattern adapted for building YAML files are presented. The proposed approaches make it possible to use additional tools to work with Kubernetes test and production clusters, which reduces the complexity of developers' interaction with them and increases deployment speed and scalability. In addition, the described methods make it possible to simplify the administration of large networks and automate the process of creating configuration YAML files for popular software templates.

Keywords: algorithm, file, configuration, class library, pattern

For citation: Fomin D. S., Balzamov A. V., Savkina A. V., Fedosin S. A., Nikulin V. V. Using multiple linear regression algorithm and design patterns to complete Kubernetes cluster configuration files. *Vestnik of Astrakhan State Technical University. Series: Management, computer science and informatics*. 2024;4:35-43. (In Russ.). https://doi.org/10.24143/2072-9502-2024-4-35-43. EDN WHFYOS.

Введение

Конфигурация кластера Kubernetes состоит из нескольких компонентов, каждый из которых имеет свои особенности настройки. Основные принципы конфигурации кластера Kubernetes [1]:

1. Иерархическая структура ресурсов: Kubernetes использует иерархическую структуру ресурсов, начиная от кластера и заканчивая отдельными контейнерами. Каждый уровень конфигурации определяет свой контекст и содержит определения других ресурсов.

2. Атомарность данных: значения в конфигурации должны быть атомарными, т. е. не должны содержать вложенные структуры данных.

3. Инверсия контекста: с помощью символа $ можно ссылаться на другие ресурсы в конфигурации, что позволяет избегать дублирования кода.

4. Валидация конфигурации: перед применением конфигурация проверяется на наличие ошибок, чтобы избежать проблем в работе кластера.

Эти принципы применимы ко всем уровням конфигурации Kubernetes, начиная от определения Pod и заканчивая настройками кластера.

YAML (YAML Ain't Markup Language) – это язык сериализации данных, который используется для описания структуры объектов и ресурсов в Kubernetes. YAML-файлы имеют определенную структуру и содержат информацию о ресурсах: их название, описание, конфигурацию и зависимости между ними [2].

При построении YAML-файла Kubernetes возможны следующие проблемы [3]:

– неправильная структура файла: YAML-файлы должны иметь определенную структуру, состоящую из уровней вложенности. Если структура файла нарушена, Kubernetes не сможет его обработать;

– ошибки в синтаксисе: YAML – это язык разметки, который требует строгого соблюдения синтаксиса. Ошибки в синтаксисе могут привести к тому, что Kubernetes не сможет прочитать файл конфигурации;

– дублирование ключей: в YAML-файлах не допускается дублирование ключей, т. к. это может привести к ошибкам в работе приложения;

– использование не валидных значений: некоторые свойства ресурсов в Kubernetes могут принимать только определенные значения. Если используется недопустимое значение, это может вызвать ошибки в работе приложения.

При изучении указанных проблем были рассмотрены проблемы конфигурации кластера и определены методы их решения с целью упрощения конфигурации кластера.

Проблема конфигурации кластера

Kubernetes состоит из двух основных объектов: Pod и Service [4].

Pod – это минимальный строительный блок в Kubernetes. Он представляет собой группу контейнеров, которые работают вместе и обеспечивают определенную функцию или сервис. Pod может содержать один или несколько контейнеров, каждый из которых выполняет свою часть приложения.

Service – это объект, который обеспечивает внешний доступ к Pod. Он создает виртуальный IP-адрес, который можно использовать для обращения к Pod внутри кластера. Service также может обеспечить балансировку нагрузки, автоматическое масштабирование и отказоустойчивость.

Существует несколько видов сервисов. Перечисленные ниже типы для простоты понимания можно рассматривать как «матрешку»: каждый последующий оборачивает предыдущий и добавляет некоторые правила маршрутизации. При создании сервиса более высокого уровня автоматически создаются сервисы нижележащего типа. Рассмотрим следующие типы сервисов [5]:

– ClusterIP – тип сервиса, существующий в любом кластере. Единая точка доступа к Pod-элементам по постоянному IP-адресу, доступному только изнутри кластера;

– NodePort – общий IP-адрес Pod-элементов (полученный из ClusterIP) соединяется с определенным портом всех Node-элементов, на которых развернуты обслуживаемые Pod-элементы. При этом Pod-элементы становятся доступны по адресу <NodeIP>:<NodePort>;

– LoadBalancer – выходной порт NodePort, который присоединяется к внешнему балансировщику нагрузки, предоставляемому облачным провайдером. Таким образом мы получаем статический внешний IP-адрес для нашего приложения.

При этом для конфигурации используются следующие объекты [6]:

– Namespace – пространство имен. Объекты могут взаимодействовать, только если находятся в одном пространстве имен. С помощью пространств имен возможно развернуть несколько виртуальных кластеров на одном физическом;

– ReplicaSet – контроллер, позволяющий создать набор одинаковых Pod-элементов и работать с ними как с единой сущностью. Он поддерживает нужное количество реплик, при необходимости создавая новые Pod-элементы или удаляя старые;

– Deployment – контроллер развертывания, являющийся абстракцией более высокого уровня над ReplicaSet'ом. Добавляет возможность обновления управляемых Pod-элементов;

– ConfigMap – объект с произвольными конфигурациями, которые могут, например, быть переданы в контейнеры через переменные среды;

– Secret – объект с некоей конфиденциальной информацией. Секреты могут быть файлами (SSL-сертификатами), которые монтируются к контейнеру, либо base64-закодированными строками, передающимися через те же переменные среды.

Для внесения небольших изменений в объекты действующего кластера можно использовать стандартную веб-консоль Kubernetes Dashboard, но для стартовой настройки всегда необходимо использовать файлы конфигурации формата YAML.

На построение синтаксически правильных файлов, как правило, уходит большое количество времени, кроме того, необходимо использовать корректные адреса репозиториев для загрузки образов контейнеров [7].

Алгоритм множественной линейной регрессии

Так как файл состоит из структурных стандартизованных блоков определенного типа, наполнение которых зависит от множества других параметров, для автоматизации процесса построения файла можно использовать алгоритм множественной линейной регрессии, применяемый при построении математических моделей для нейронных сетей. Воспользуемся уравнением множественной линейной регрессии, которое имеет следующий вид:

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_j x_j + \varepsilon,$$

где y – зависимая переменная; θ_0 – коэффициент, задающий базовый уровень; $\theta_1, \theta_2, \ldots, \theta_j$ – коэффициенты изменения зависимой переменной; $x_1, x_2, \ldots, x_j$ – независимые переменные; ε – коэффициент отклонения фактических данных от прогнозируемых.

В отличие от простой линейной регрессии в данном случае несколько признаков x (независимых переменных) и несколько коэффициентов θ.

Коэффициент θ_0 задает некоторый базовый уровень при условии, что остальные коэффициенты равны нулю и зачастую не имеют смысла с точки зрения интерпретации модели.

Параметры $\theta_1, \theta_2, \ldots, \theta_j$ показывают изменение зависимой переменной при условии «неподвижности» остальных коэффициентов.

Переменная ε (ошибка) представляет собой отклонение фактических данных от прогнозируемых. В этой переменной заложены две составляющие. Во-первых, она может включать вариативность целевой переменной, описанную другими (не включенными в указанную модель) признаками. Во-вторых, «улавливать» случайный шум, случайные колебания [8].

В предложенном алгоритме будет использована выборка данных на основе опыта пользователей. Кроме того, будут заданы известные стартовые эталонные данные для корректной работы алгоритма, представленные на рис. 1.

	Value	SourceLinkId	KubernetesModuleTypeId	KubernetesObjectTypeId
4	1.00000000	1	7	7
5	1.00000000	2	6	1
6	0.30000000	2	2	3
7	0.50000000	2	1	4
8	0.90000000	7	6	1
9	0.70000000	7	6	2
10	0.93000000	8	6	1
11	0.73000000	8	6	2
12	0.40000000	9	6	1
13	0.20000000	9	2	2
14	0.40000000	3	1	1
15	0.34000000	10	1	1
16	0.10000000	10	5	2
17	0.90000000	11	1	1
18	0.23000000	11	3	2
19	0.70000000	12	1	1
20	0.10000000	12	4	2
21	0.70000000	5	2	1
22	0.20000000	13	6	1

Рис. 1. Пример эталонных данных
для одного из шаблонов

Fig. 1. Example of reference data
for one of the templates

Математическая модель содержит в себе следующие переменные:

– идентификатор справочника типов объекта Kubernetes (Pod, Service и т. д.);

– идентификатор справочника ссылок на репозиторий Docker контейнера;

– идентификатор справочника типов модулей, используемых объектами Kubernetes, который представляет собой значение справочника модулей объектов Kubernetes, использующихся при создании кластера;

– зависимая переменная, вычисляемая для каждой записи в массиве данных.

На рис. 2 представлен пример файла Kubernetes с соответствием переменных предикторов с реальными сущностями файла.

```
mple.yaml
apiVersion: apps/v1
kind: Deployment #Переменная предиктор
metadata:
  name: nginx
  labels:
    app: nginx #Переменная предиктор
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers: #Переменная предиктор
      - name: nginx
        image: nginx:latest
        ports:
        - containerPort: 80
```

Рис. 2. Пример файла конфигурации Kubernetes

Fig. 2. Example of Kubernetes configuration file

Перед применением алгоритма множественной линейной регрессии следует убедиться, что данные можно аппроксимировать с помощью линейной модели.

Для проверки линейной зависимости достаточно построить диаграмму рассеяния для каждой переменной-предиктора и переменной-ответа. Диаграмма рассеяния, построенная с помощью встроенных инструментов языка программирования R, представлена на рис. 3.

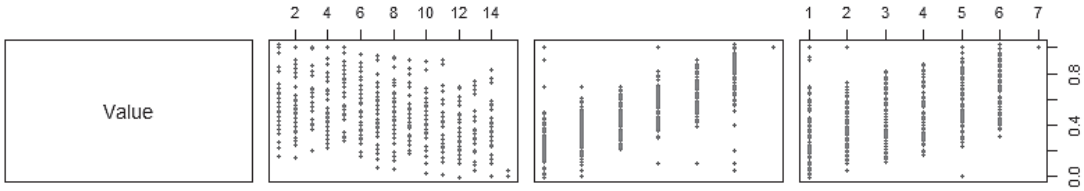

Рис. 3. Диаграмма рассеяния

Fig. 3. Scattering diagram

Как видно из диаграммы, каждая переменная-предиктор имеет линейную корреляцию с переменной отклика.

Далее можно проверить распределение невязок модели, которое должно быть нормальным. Для этого построим гистограмму остатков математической модели, представленной на рис. 4.

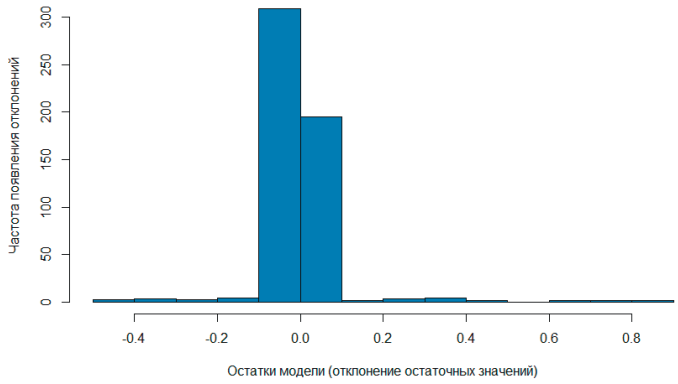

Рис. 4. Гистограмма остатков

Fig. 4. Histogram of residuals

Хотя распределение немного смещено вправо, оно не настолько ненормально, чтобы вызывать серьезные опасения.

Также нужно проверить выполнение условия гомоскедастичности. Чтобы проверить, соблюдается ли это предположение, построим график зависимости подходящего значения от остатка. График представлен на рис. 5.

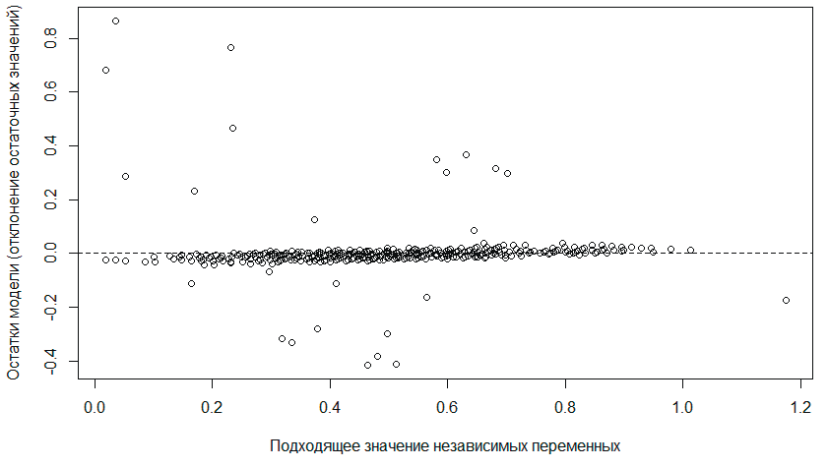

Рис. 5. График зависимости подходящего значения от остатка

Fig. 5. Graph of the dependence of the appropriate value on the remainder

Как видно из графика, разброс имеет тенденцию к увеличению при малых подобранных значениях, но эта закономерность не вызывает большого беспокойства.

Для реализации алгоритма множественной линейной регрессии необходима стартовая матрица, состоящая из набора данных, описанного ранее. При этом набор данных запрашивается из заранее подготовленной базы данных по идентификатору справочника типа шаблона Kubernetes. Алгоритм состоит из следующих шагов [9]:

1. Исходная матрица транспонируется.
2. Транспонированная матрица перемножается с исходной.
3. Перемноженная матрица инвертируется.
4. Инвертированная матрица перемножается с транспонированной.
5. Из матрицы извлекается вектор коэффициента b для линейного уравнения $y = b_0 x_0 + b_1 x_1 + b_2 x_2 + \ldots + b_n x_n$, где b_n – коэффициенты изменения зависимой переменной; x_n – независимые переменные на основе выбранных предикторов и зависимой переменной.
6. На основании извлеченного вектора b вычисляется зависимая переменная y.

Полученные значения зависимой переменной обновляются в базе данных для всех отобранных записей и затем выбираются уникальные записи с самым большим коэффициентом для требуемого шаблона. Затем обработанные данные используем для построения конфигурационного файла Kubernetes.

Результирующий набор данных

В результате анализа массива данных с помощью алгоритма множественной линейной регрессии, собранных на основе опыта пользователей через web-приложение конструктора YAML-файлов, были составлены наборы объектов Kubernetes для разных шаблонов кластеров. Для каждого шаблона было взято около 500 оценочных значений в виде, представленном на рис. 6.

	Value	SourceLinkId	KubernetesModuleTypeId	KubernetesObjectTypeId
29	0.43915528	5	1	6
30	0.19982935	12	1	4
31	0.43800394	8	4	2
32	0.73082434	7	6	3
33	0.45859599	2	4	1
34	0.52048960	2	2	5
35	0.21900197	4	2	1
36	0.22068949	11	3	1
37	0.40238011	10	2	5
38	0.64224222	13	6	3
39	0.43942337	10	3	4
40	0.20592581	12	3	1
41	0.24852934	2	2	1
42	0.47037017	10	2	6
43	0.65673782	7	4	5
44	0.43190748	8	2	5
45	0.67874930	3	3	6
46	0.68484576	3	5	3
47	0.38450947	2	2	3

Рис. 6. Часть массива данных опыта пользователей

Fig. 6. Part of the user experience data set

Столбец Value содержит рассчитанное значение значимой переменной. Столбцы SourceLinkId, KubernetesModuleTypeId и KubernetesObjectTypeId содержат значения переменных-предикторов.

Были отобраны записи с наибольшими значимыми коэффициентами. Шаблоны и наборы объектов представлены в таблице.

Выходные данные алгоритма

The output of the algorithm

Шаблон	Тип объекта	Репозиторий Docker Hub	Тип модуля объекта	Значение зависимой переменной
SPA	Deployment	Alpine https://hub.docker.com/_/alpine	OS	0,69
	Service	NGINX https://hub.docker.com/_/nginx	Web server	0,505
ASP NET Core WebApp SSR	Deployment	https://hub.docker.com/_/microsoft-dotnet-aspnet	OS	0,73
	Service	NGINX https://hub.docker.com/_/nginx	Web server	0,47
	StatefulSet	Postgres https://hub.docker.com/_/postgres	DB	0,39
	Secret	–	API keys	0,87
	Service	NextJS https://hub.docker.com/r/richardkovacs/nextjs	SSR Web server	0,45
DB Cluster	StatefulSet	Postgres https://hub.docker.com/_/postgres	Master DB	0,92
			Slave DB	0,64
				0,33

Для каждого шаблона отобраны определенные наборы модулей, без которых полноценное функционирование невозможно. Для кластера баз данных (шаблон DB Cluster) необходимы хотя бы один мастер-экземпляр и несколько зависимых. Одностраничному приложению (шаблон SPA) достаточно легковесного Linux дистрибутива Alpine и веб-сервера NGINX для доступа извне. Самым требовательным является полноценное веб-приложение (шаблон ASP NET Core WebApp SSR). Для него была подобрана среда с предустановленным пакетом SDK ASP NET Core, веб-сервер NGINX, база данных Postgres, сервис хранения API-ключей и веб-сервер NextJS для технологии серверного рендеринга контента страницы (SSR) [10]. По оценкам опыта пользователей указанные наборы объектов используют открытый исходный код и являются минимальными и достаточными.

Реализация сервиса построения конфигурационного YAML-файла для Kubernetes

Чтобы интерпретировать полученные данные после обработки алгоритмом в конечный файл конфигурации YAML, можно воспользоваться паттерном «Компоновщик» [11].

Паттерн состоит из следующих компонентов:

– Component: определяет интерфейс для всех компонентов в древовидной структуре;

– Composite: представляет компонент, который может содержать другие компоненты и реализует механизм для их добавления и удаления;

– Leaf: представляет отдельный компонент, который не может содержать другие компоненты;

– Client: клиент, который использует компоненты.

В упрощенной схеме паттерна реализацию компонента Leaf можно опустить при условии, что использоваться будут только составные иерархические объекты [12].

Как было сказано ранее, YAML-файл представляет собой иерархическую структуру составных объектов, что полностью совпадает со спецификой и областью применения паттерна «Компоновщик».

После выполнения алгоритма множественной линейной регрессии и обновления весовых коэффициентов на основе зависимой переменной из базы данных выбираются уникальные записи с самым большим коэффициентом для требуемого шаблона и передаются в сервис построения конфигурационных файлов Kubernetes. Данный сервис, применяя паттерн «Компоновщик» на основе входных записей требуемых объектов для данного шаблона, собирает конечный YAML-файл для дальнейшего его применения на Kubernetes кластере. Сервис реализован на языке C#. Схема работы реализованного сервиса представлена в виде диаграммы последовательности на рис. 7.

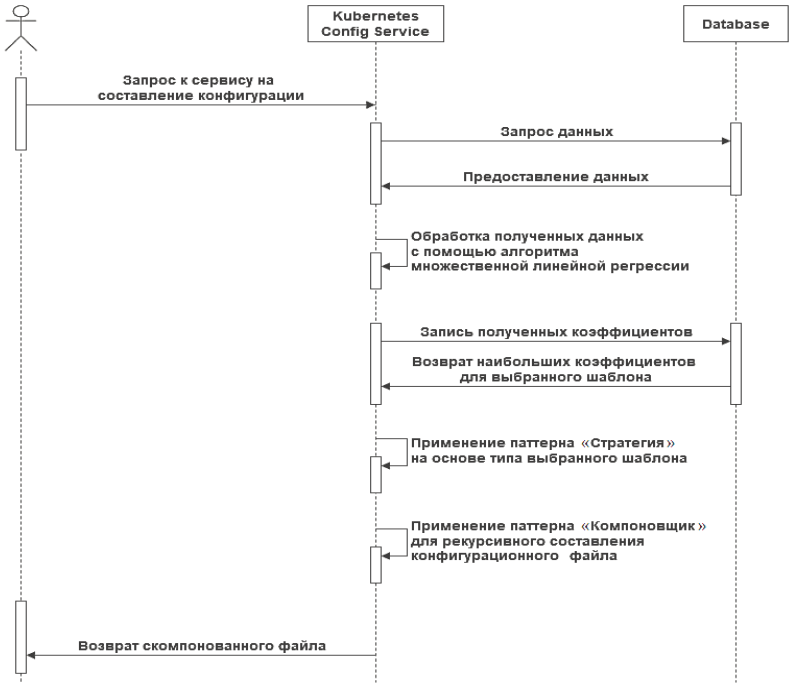

Рис. 7. Схема работы реализованного сервиса

Fig. 7. Scheme of operation of the implemented service

В диаграмме отражен основной бизнес-процесс сервиса построения конфигурационного YAML-файла для Kubernetes:

1. Пользователь отправляет http-запрос к сервису на формирование конфигурационного сервиса по требуемому шаблону.

2. Сервис запрашивает данные из реляционной СУБД, содержащей статистические и структурные данные.

3. Найденные данные возвращаются сервису.

4. Полученные данные обрабатываются с помощью алгоритма множественной линейной регрессии.

5. Рассчитанные коэффициенты записываются в СУБД.

6. СУБД возвращает наибольшие коэффициенты для обрабатываемого типа шаблона.

7. Применяется паттерн «Стратегия» для вызова класса в соответствии с типом шаблона.

8. Применяется паттерн «Компоновщик» для рекурсивного построения конфигурационного файла на основе коэффициентов, полученных в шаге 6.

9. Скомпонованный файл возвращается пользователю.

Заключение

Таким образом, используя алгоритм множественной линейной регрессии, можно добиться автоматизации процесса построения кластеров Kubernetes. Описанный метод позволит снизить временные и человеческие ресурсы, но при этом возрастут требования к производительности сервера, где будут происходить алгоритмические вычисления. При этом важно отметить, что затраты на вычислительные мощности всегда ниже остальных. К тому же, с постоянно обновляющимся массивом данных, основанным на опыте пользователей, сервис обеспечит возможность проводить аналитику удачных решений и применять их в будущем.

Описанный метод оптимизации может быть применен в различных сферах деятельности, где используются механизмы Kubernetes, без привязки к специфике и области работы, будь то образовательное учреждение либо коммерческое или государственное предприятие, т. е. метод является универсальным и эффективным средством построения кластеров Kubernetes.

Приведенный метод для построения конфигурационных YAML-файлов для Kubernetes позволяет автоматизировать процесс построения YAML-файла Kubernetes, который сводится к подготовке статистических обработанных данных и дальнейшей их интерпретации. Описанный метод построения файлов позволит снизить временные и человеческие ресурсы, кроме этого, возможность ошибки при построении кластера существенно снизится.

Сервис компоновщика файлов может быть использован и для построения других форматов файлов, имеющих как иерархическую, так и flat (плоскую) структуру.

Список источников

1. Основы Kubernetes. URL: https://habr.com/ru/articles/258443/ (дата обращения: 23.10.2023).

2. Objects In Kubernetes // Kubernetes. URL: https://kubernetes.io/docs/concepts/overview/working-with-objects (дата обращения: 23.10.2023).

3. Managing YAML Errors for Kubernetes Configuration. URL: https://dev.to/olabayobalogun/managing-yaml-errors-for-kubernetes-configuration-k85 (дата обращения: 23.10.2023).

4. Balzamov A. V., Fomin D. S., Savkina A. V., Nikulin V. V., Fedosin S. A. Development of a methodology for migration of monolithic systems to micro-service architecture using cloud technologies // Journal of Physics: Conference Series. 2021. 5. Сер.: 5th International Scientific Conference on Information, Control, and Communication Technologies, ICCT 2021. С. 012036.

5. Фомин Д. С., Савкина А. В. Проблемы реализации монолитных систем // XLIX Огаревские чтения: материалы Науч. конф.: в 3 ч. Саранск: Национ. исслед. Мордов. гос. ун-т им. Н. П. Огарёва, 2021. С. 257–263.

6. Namespaces // Kubernetes. URL: https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/ (дата обращения: 24.10.2023).

7. Маркелов А. Введение в технологии контейнеров и Kubernetes. М.: ДМК Пресс, 2019. С. 110–152.

8. Множественная линейная регрессия. URL: https://www.dmitrymakarov.ru/opt/mlr-04/#9-3-gomoskedastichnost-ostatkov (дата обращения: 25.10.2023).

9. Модель множественной линейной регрессии: содержательная интерпретация и предпосылки. URL: https://sun.tsu.ru/mminfo/2016/Dombrovski/book/chapter-3/chapter-3-1.htm (дата обращения: 25.10.2023).

10. Дейтрей С., Харрелл Ф. Линейная регрессия с примерами приложений. М.: Вильямс, 2018. 800 с.

11. Гамма Э., Хелм Р., Джонсон Р. Паттерны объектно-ориентированного проектирования. СПб.: Питер, 2021. 142 с.

12. Фримен Э., Робсон Э., Сьерра К. Паттерны проектирования. СПб.: Питер, 2021. 640 с.

References

1. *Osnovy Kubernetes*. Available at: https://habr.com/ru/articles/258443/ (accessed: 23.10.2023).

2. Objects In Kubernetes. *Kubernetes*. Available at: https://kubernetes.io/docs/concepts/overview/working-with-objects/ (accessed: 23.10.2023).

3. *Managing YAML Errors for Kubernetes Configuration*. Available at: https://dev.to/olabayobalogun/managing-yaml-errors-for-kubernetes-configuration-k85 (accessed: 23.10.2023).

4. Balzamov A. V., Fomin D. S., Savkina A. V., Nikulin V. V., Fedosin S. A. Development of a methodology for migration of monolithic systems to micro-service architecture using cloud technologies. *Journal of Physics: Conference Series. 2021. 5. Seriia: 5th International Scientific Conference on Information, Control, and Communication Technologies, ICCT 2021*. P. 012036.

5. Fomin D. S., Savkina A. V. Problemy realizatsii monolitnykh sistem [Problems of implementation of monolithic systems]. *XLIX Ogarevskie chteniia: materialy Nauchnoi konferentsii: v 3 ch*. Saransk, Natsion. issled. Mordov. gos. un-t im. N. P. Ogareva, 2021. Pp. 257-263.

6. *Namespaces. Kubernetes*. Available at: https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/ (accessed: 24.10.2023).

7. Markelov A. *Vvedenie v tekhnologii konteinerov i Kubernetes* [Introduction to Container and Kubernetes Technologies]. Moscow, DMK Press, 2019. Pp. 110-152.

8. *Mnozhestvennaia lineinaia regressiia* [Multiple linear regression]. Available at: https://www.dmitrymakarov.ru/opt/mlr-04/#9-3-gomoskedastichnost-ostatkov (accessed: 25.10.2023).

9. *Model' mnozhestvennoi lineinoi regressii: soderzhatel'naia interpretatsiia i predposylki* [Multiple linear regression model: meaningful interpretation and background]. Available at: https://sun.tsu.ru/mminfo/2016/Dombrovski/book/chapter-3/chapter-3-1.htm (accessed: 25.10.2023).

10. Deitrei S., Kharrell F. *Lineinaia regressiia s primerami prilozhenii* [Linear regression with application examples]. Moscow, Vil'iams Publ., 2018. 800 p.

11. Gamma E., Khelm R., Dzhonson R. *Patterny ob"ektno-orientirovannogo proektirovaniia* [Object-oriented design patterns]. Saint Petersburg, Piter Publ., 2021. 142 p.

12. Frimen E., Robson E., S'erra K. *Patterny proektirovaniia* [Design patterns]. Saint Petersburg, Piter Publ., 2021. 640 p.

Статья поступила в редакцию 10.04.2024; одобрена после рецензирования 05.10.2024; принята к публикации 11.10.2024
The article was submitted 10.04.2024; approved after reviewing 05.10.2024; accepted for publication 11.10.2024

Информация об авторах / Information about the authors

Дмитрий Сергеевич Фомин – аспирант кафедры автоматизированных систем обработки информации и управления; Национальный исследовательский Мордовский государственный университет им. Н. П. Огарёва; nkfominsa@mail.ru

Dmitry S. Fomin – Postgraduate Student of the Department of Automated Information Processing and Control Systems; National Research Ogarev Mordovia State University; nkfominsa@mail.ru

Александр Витальевич Бальзамов – аспирант кафедры автоматизированных систем обработки информации и управления; Национальный исследовательский Мордовский государственный университет им. Н. П. Огарёва; veron13rus@yandex.ru

Анастасия Васильевна Савкина – кандидат технических наук, доцент; доцент кафедры автоматизированных систем обработки информации и управления; Национальный исследовательский Мордовский государственный университет им. Н. П. Огарёва; av-savkina@yandex.ru

Сергей Алексеевич Федосин – кандидат технических наук, профессор; заведующий кафедрой автоматизированных систем обработки информации и управления; Национальный исследовательский Мордовский государственный университет им. Н. П. Огарёва; fedosinsa@mrsu.ru

Владимир Валерьевич Никулин – кандидат технических наук, доцент; заведующий кафедрой инфокоммуникационных технологий и систем связи; Национальный исследовательский Мордовский государственный университет им. Н. П. Огарёва; nikulinvv@mail.ru

Alexander V. Balsamov – Postgraduate Student of the Department of Automated Information Processing and Control Systems; National Research Ogarev Mordovia State University; veron13rus@yandex.ru

Anastasia V. Savkina – Candidate of Technical Sciences, Assistant Professor; Assistant Professor of the Department of Automated Information Processing and Control Systems; National Research Ogarev Mordovia State University; av-savkina@yandex.ru

Sergey A. Fedosin – Candidate of Technical Sciences, Professor; Head of the Department of Automated Information Processing and Control Systems; National Research Ogarev Mordovia State University; fedosinsa@mrsu.ru

Vladimir V. Nikulin – Candidate of Technical Sciences, Assistant Professor; Head of the Department of Information and Communication Technologies and Communication Systems; National Research Ogarev Mordovia State University; nikulinvv@mail.ru