УДК 004.41
DOI 10.47367/0021-3497_2024_5_191

АВТОМАТИЗАЦИЯ ПРОЦЕССА РАЗВЕРТЫВАНИЯ ПРОГРАММНЫХ ТЕХНИЧЕСКИХ СРЕДСТВ СИСТЕМ МОНИТОРИНГА СОСТОЯНИЯ ОБОРУДОВАНИЯ ТЕКСТИЛЬНОГО ОТДЕЛОЧНОГО ПРОИЗВОДСТВА И ЕГО ДИСПЕТЧЕРИЗАЦИИ

AUTOMATION OF THE DEPLOYMENT PROCESS OF SOFTWARE AND TECHNICAL TOOLS FOR MONITORING THE CONDITION OF EQUIPMENT IN THE TEXTILE FINISHING PRODUCTION AND ITS DISPATCHING

П.Г. ФРАСЫН, С.Л. ВЛАСОВ, Е.А. РЫЖКОВА

P.G. FRASYN, S.L. VLASOV, E.A. RYZHKOVA

(Российский государственный университет имени А.Н. Косыгина (Технологии. Дизайн. Искусство))

(The Kosygin State University of Russia)

E-mail: frasyn.pashka@gmail.com

Рассмотрены методы организации системы мониторинга состояния оборудования текстильного отделочного производства и его диспетчеризации. Изучены и проанализированы технологии взаимодействия компонентов системы мониторинга оборудования текстильного отделочного произ-

водства. Особое внимание уделено архитектурным подходам к динамической конфигурации и развертыванию системы с применением Docker, Ansible и ролевой организации компонентов. Результаты исследования доказали универсальность механизма конфигурации для обеспечения гибкости и масштабируемости системы. Оптимизация архитектуры и выбор подходящих технологий взаимодействия компонентов системы способствуют устойчивости и безотказной работе системы даже в условиях повышенной нагрузки или непредвиденных сбоев. Полученные результаты предоставляют основу для разработки и внедрения современных систем мониторинга и диспетчеризации не только для оборудования текстильного отделочного производства, но и в промышленной сфере в целом, способствуя повышению производительности и надежности процессов управления оборудованием. разработка и применение надежных методов управления конфигурацией и обеспечения стабильности работы системы является критически важным аспектом для успешной эксплуатации технологических объектов.

This study examines methods for organizing a monitoring and dispatching system for the condition of equipment in the textile finishing production. The technologies for interaction between the components of the monitoring system and their application for textile finishing equipment were studied and analyzed. Special attention was given to architectural approaches to dynamic configuration and deployment of the system using Docker, Ansible, and the role-based organization of components. The research results demonstrated the versatility of a universal configuration mechanism to ensure system flexibility and scalability. Optimizing architecture and selecting appropriate technologies for component interaction contribute to the system's stability and reliable operation even under conditions of increased load or unforeseen failures.

The obtained results provide a foundation for the development and implementation of modern monitoring and dispatching systems not only for equipment in textile finishing production but also in the industrial sector as a whole, contributing to the enhancement of productivity and reliability in equipment management processes. Thus, the development and application of reliable configuration management methods and ensuring system stability are critically important aspects for the successful operation of technological objects.

Ключевые слова: система мониторинга, промышленные объекты, динамическая конфигурация, ролевая организация, надежность, масштабируемость, процесс развертывания, управление конфигурацией.

Keywords: monitoring system, industrial objects, dynamic configuration, role-based organization, reliability, scalability, deployment process, configuration management.

Введение

На текущий момент автоматизация современного текстильного отделочного производства не может обойтись без систем мониторинга и диспетчеризации. Они играют ключевую роль в повышении эффективности производства, снижении затрат и улучшении безопасности.

С помощью систем мониторинга осуществляется наблюдение за различными параметрами и состоянием оборудования в реальном времени. Системы диспетчеризации позволяют централизованно управлять производственным процессом, контролировать работу оборудования, распределять ресурсы и оптимизировать производствен-

ный процесс в целом, что в совокупности дает операторам возможность быстро реагировать на изменения и координировать действия для обеспечения эффективной работы предприятия.

Вопрос надежности данных систем является критически важным, особенно в контексте сложности текстильного отделочного производства, где даже кратковременные сбои могут привести к серьезным последствиям: простоям оборудования, потери продукции, а также создавать угрозы для безопасности персонала и окружающей среды.

Анализ ряда исследований [1…3], проведенных в области применения информационных технологий, свидетельствует, что для повышения качества программных технических средств требуется использование как передовых технологий и алгоритмов, так и комплексного подхода к разработке программного обеспечения, основанного на интеграции систем контроля версий Git, с непрерывной интеграцией (CI) и непрерывной доставкой (CD).

Основой CI/CD систем является контейнеризация программного обеспечения. Она гарантирует согласованность среды – приложение будет развернуто в точно такой же среде, в которой оно было протестировано и развернуто на других этапах процесса. Контейнеры обеспечивают легкую масштабируемость, позволяя запускать множество экземпляров контейнеров одного и того же приложения для обработки больших объемов работы, а также позволяют управлять зависимостями и версиями приложений путем включения всех необходимых компонентов и библиотек в контейнер. Это облегчает управление зависимостями и предотвращает конфликты версий [4].

Основная часть

Программное обеспечение собирается в Docker образ, после чего хранится в Docker registry, которое представляет собой централизованное хранилище для контейнеризированных образов (рис. 1), управляемых и доступных для загрузки с помощью произвольных меток версий, таких как "1.0.0", "1.0.2", "1.0.3", "ver1".

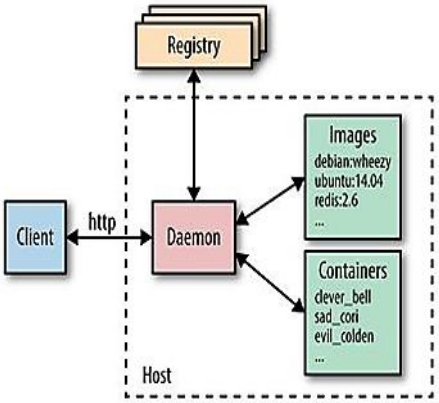

Рис. 1

Материалы и методы

В данной работе рассматривается упрощенная система мониторинга и диспетчеризации, содержащая часть необходимых сервисов, каждый из которых упакован в docker образ и загружен в registry.

Сервис агрегации данных является центральным узлом системы, ответственным за сбор, обработку и передачу данных, поступающих от различных устройств и оборудования на промышленном объекте. Этот компонент обеспечивает передачу получаемой информации на следующие этапы обработки.

Сервис сервера базы данных представляет собой хранилище данных, обеспечивающее постоянный доступ к информации, собранной агрегатором данных. Он обладает механизмами управления данными, включая индексацию, архивацию и обеспечение безопасности хранимых данных.

Сервис брокера данных является промежуточным звеном (шлюзом) между агрегатором данных и другими компонентами системы. Его основная функция состоит в маршрутизации и фильтрации данных, а также обеспечении асинхронной связи между различными частями системы.

Визуальный интерфейс системы диспетчеризации и мониторинга обеспечивает пользовательский доступ к данным, собранным и обработанным системой. Он предоставляет средства визуализации, анализа и управления данными, позволяя операторам эффективно управлять промышленным процессом.

Каждый из сервисов должен предоставлять набор API (интерфейсов программирования приложений) или протоколов коммуникации, которые определяют способы передачи данных и команд между сервисами. Эти интерфейсы могут быть основаны на стандартных протоколах связи, таких как HTTP, MQTT, AMQP и другие, либо могут быть специально разработаны в соответствии с потребностями системы. Использование интерфейсов позволяет обеспечить гибкость и расширяемость системы, легко добавлять новые сервисы или модифицировать существующие без необходимости полной переработки архитектуры системы [5].

Агрегатор данных выполняет сбор информации с оборудования и направляет ее на брокер сообщений посредством протокола MQTT, который затем передает данные в сервис сервера базы данных для хранения и обработки. Визуальный интерфейс системы в свою очередь использует веб-сокеты для обновления данных в реальном времени и получает доступ к информации через сервис сервера базы данных, обеспечивая операторам возможность эффективного мониторинга и управления промышленным процессом.

В контексте неопределенности относительно конкретной конфигурации установки системы, состоящей из различных сервисов, наша задача заключается в разработке универсального механизма конфигурации, способного динамически формировать соединения между компонентами в зависимости от текущих потребностей. Данная проблематика требует разработки гибкой архитектуры, которая позволит автоматизировать процесс установки и настройки системы с учетом разнообразных комбинаций сервисов и их взаимосвязей. В рамках этого подхода необходимо определить набор параметров конфигурации, описывающих требуемые связи между сервисами, а также механизмы их автоматического создания и управления. Предполагается использование современных технологий виртуализации, контейнеризации и оркестрации, таких как Docker, Kubernetes и Ansible, для реализации динамического механизма конфигурации, способного обеспечить гибкость и масштабируемость системы в условиях изменяющихся требований и условий эксплуатации [6].

Важным аспектом данного подхода является обеспечение надежности и безопасности конфигурации, а также ее возможности расширения и адаптации к различным сценариям использования. Такой подход позволит эффективно решать задачу развертывания и управления системой мониторинга и диспетчеризации в различных промышленных средах [11].

Для облегчения процесса создания и доставки файлов деплоя (docker compose) на целевой сервер в нашем исследовании было принято решение использовать программный продукт с открытым исходным кодом Ansible. Он является системой управления конфигурациями, основанной на концепциях ролей и плейбуков. Роль в Ansible представляет собой набор действий, которые будут выполнены на целевой машине. Она позволяет абстрагировать и организовать логику деплоя, облегчая масштабирование и поддержку инфраструктуры [7]. Плейбук является главным файлом, в котором описывается набор действий, предпринимаемых на основе ролей.

Используя Docker и Docker Compose, мы можем динамически генерировать файлы docker-compose в соответствии с требованиями и зависимостями используемых сервисов. Это позволит нам автоматизировать процесс развертывания и масштабирования инфраструктуры в зависимости от изменяющихся потребностей.

Используя Ansible, интегрируем следующую логику формирования docker-compose файлов в плейбуки и роли [8, 9].

В случае необходимости взаимодействия между сервисами, например, между сервисом агрегации данных и сервисом сервера базы данных необходимо определить механизм интеркоммуникации (intercomm), который, будучи объявленным в плейбуке, автоматически подключится к интерфейсу зависимого сервиса. Этот механизм позволит сервису обращаться к другим сервисам через определенные прото-

колы и интерфейсы, обеспечивая эффективное взаимодействие между компонентами системы [10].

В контексте данной статьи мы не рассматриваем подробную работу физических и виртуальных сетей, а лишь описываем механизмы коммуникации в упрощенном виде.

Результаты и их обсуждение

Интеркоммуникация (intercomm) ведущего сервиса может быть передана в качестве параметра или объекта внедрения зависимостей (dependency injection) в зависимый сервис. Для этого необходимо корректно настроить внедрение зависимостей в зависимом сервисе, так чтобы он мог принимать и использовать интеркоммуникацию от ведущего сервиса.

Таким образом, взаимодействие между сервисами достигается за счет описания логики коммуникации вторым уровнем вложенности в Ansible.

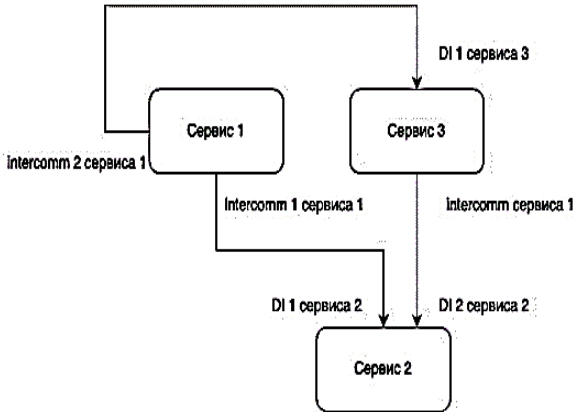

Рис. 2

На рис. 2 представлена схема взаимодействия трех сервисов. Сервис 1 обладает двумя интерфейсами интеркоммуникации, которые служат объектами внедрения зависимостей в сервисы 3 и 2. Сервис 3 обладает собственным интерфейсом, который предоставляется в качестве внедрения зависимости в сервис 2. Такая архитектура обеспечивает легкую масштабируемость и транспортабельность системы, поскольку каждый сервис изолирован и имеет четко определенные интерфейсы взаимодействия. Это позволяет гибко добавлять новые сервисы или модифицировать существующие, не затрагивая другие компоненты системы. Универсальный механизм конфигурации остается критически важным, поскольку он позволяет адаптировать связи между компонентами системы в соответствии с изменяющимися потребностями и условиями эксплуатации. Он может включать в себя динамическое создание, изменение или удаление связей между сервисами, а также настройку параметров соединений в зависимости от текущих требований.

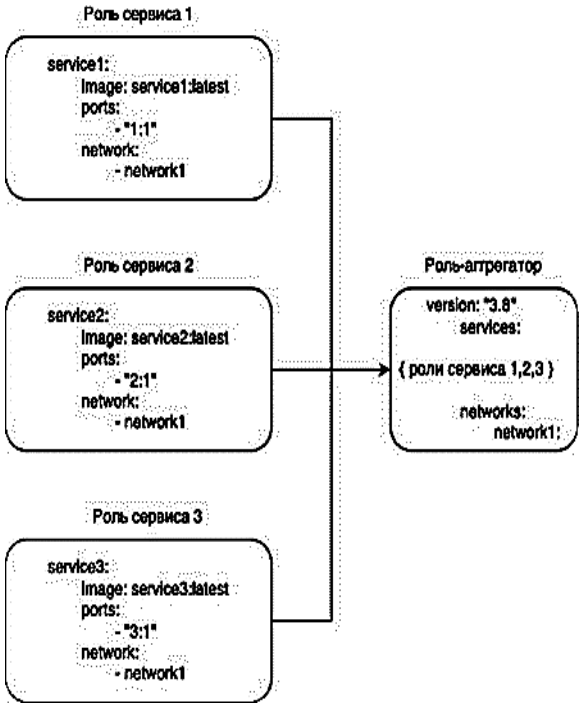

Рис. 3

Использование compose (или кластерного docker swarm) для развертывания набора сервисов в рамках либо одной инсталляции, либо кластерно-распределенной требует наличия сконфигурированного файла docker compose. Для создания такого файла динамически, основываясь на взаимодействии сервисов, мы разработали концепцию генерации развертывающего конфигурационного docker compose файла, где каждый сервис представлен в виде отдельной роли, содержащей часть сервиса (services).

В контексте автоматизации процесса развертывания главной заслугой Ansible является способность эффективно управлять удаленными системами посредством простого протокола SSH, обеспечивающего возможность развертывания и конфигура-

ции ПО на целевых серверах путем выполнения плейбуков, которые содержат инструкции для установки, настройки и запуска необходимых сервисов и приложений.

В ходе выполнения плейбука Ansible последовательно решает задачи, определенные в ролях, включенных в этот плейбук, обеспечивая корректное развертывание компонентов инфраструктуры. Каждая роль осуществляет свою часть конфигурации, что способствует управлению и согласованности процесса развертывания. После завершения выполнения плейбука Ansible предоставляет информацию о статусе операций и завершает процесс развертывания, тем самым обеспечивая эффективное управление конфигурацией системы в рамках заданных параметров и требований.

ВЫВОДЫ

Рассмотрены ключевые аспекты разработки и управления системой мониторинга и диспетчеризации текстильного отделочного производства. Обсуждены различные методы взаимодействия компонентов системы, включая использование различных технологий передачи данных.

Проанализированы архитектурные подходы к динамической конфигурации и развертыванию системы с использованием инструментов, таких как Docker Compose, Ansible и ролевая организация используемых сервисов.

Предложен универсальный механизм конфигурации для обеспечения гибкости и масштабируемости системы, а также рассмотрены вариации распределенных и локальных развертываний программных технических средств.

Представлен обзор процесса развертывания плейбука из полученных ролей с использованием Ansible, выделены ключевые шаги и методы для управления конфигурацией и обеспечения стабильности и надежности работы системы.

Все описанные аспекты подчеркивают значимость системного подхода к проектированию системы мониторинга и диспетчеризации и управлению ею, что позволяет обеспечить эффективное и надежное ее функционирование.

ЛИТЕРАТУРА

1. *Никифоров А.В.* Как внедрение непрерывной интеграции и тестирование помогает обеспечить соответствие требованиям при разработке программного продукта // International journal of Professional Science. 2023. №4. С. 88…96.
2. *Карпов В.В., Карпов А.В.* Особенности применения современных методов разработки программного обеспечения защищенных автоматизированных систем // Программные продукты и системы. 2016. №1. С. 5…12.
3. *Литвяков Д.С.* Разработка конфигурации CI/CD для автоматизации развертывания и управления приложениями // Научно-образовательный журнал для студентов и преподавателей «StudNet». 2021. №6. С. 1089…1097.
4. *Лазарева Н.Б.* Автоматизация развертывания Kubernetes-кластеров на базе Ubuntu ОС в Rancher на инфраструктуре VMWare vSphere // Инженерный вестник Дона. 2023. №4.
5. *Гумеров Б.З.* Автоматизация развертывания геораспределенных кластеров SPLUNK с помощью TERRAFORM и ANSIBLE // Universum: Технические науки. 2022. №5.
6. *Беспятов М.В.* Автоматизация и оптимизация процессов разработки и развертывания в DevOps: применение современных методов и инструментов // Инновации и инвестиции. 2023. №7. С. 458…464.
7. *Грушин Д.А., Кузюрин Н.Н.* Задачи оптимизации размещения контейнеров MPI-приложений на вычислительных кластерах // Труды ИСП РАН. 2017. №6. С. 458…464.
8. *Брыжинская А.В.* Кратко про Docker // Теория и практика современной науки. 2019. №10. С. 33…35.
9. *Емельянова С.С., Иващенко Н.Н.* Исследование Docker в части контейнеризации приложений на уровне ОС // Научно-технический вестник Поволжья. 2021. №12. С. 149…151.
10. *Ермаков А.С.* Перспективное развитие методологии DevOps // Вестник НГУЭУ. 2020. №4. С. 174…183.
11. *Лукашенко М.А., Телегина Т.В.* Управление созданием образовательных продуктов с помощью метода Scrum // Азимут научных исследований: экономика и управление. 2019. №2. С. 223…227.

REFERENCES

1. *Nikiforov A.V.* How the implementation of continuous integration and testing helps ensure compliance with requirements in software product development // International Journal of Professional Science. 2023. №4. pp. 88…96.

2. *Karpov V.V., Karpov A.V.* Features of applying modern methods of software development in secure automated systems // Software Products and Systems. 2016. №1. pp. 5…12.

3. *Litvyakov D.S.* Development of CI/CD configuration for automation of application deployment and management // Scientific and Educational Journal for Students and Teachers "StudNet". 2021. №6. pp. 1089…1097.

4. *Lazareva N.B.* Automation of deploying Kubernetes clusters based on Ubuntu OS in Rancher on VMWare vSphere infrastructure // Engineering Herald of Don. 2023. №4.

5. *Gumerov B.Z.* Automation of deploying geodistributed SPLUNK clusters using TERRAFORM and ANSIBLE // Universum: Technical Sciences. 2022. №5.

6. *Bespyatov M.V.* Automation and optimization of development and deployment processes in DevOps: application of modern methods and tools // Innovations and Investments. 2023. №7. pp. 458…464.

7. *Grushin D.A., Kuzyurin N.N.* Optimization tasks for placing MPI application containers on computing clusters // Proceedings of ISP RAS. 2017. №6. pp. 458…464.

8. *Bryzhinskaya A.V.* Briefly about Docker // Theory and Practice of Modern Science. 2019. №10. pp. 33…35.

9. *Emelyanova S.S., Ivashchenko N.N.* Study of Docker in terms of application containerization at the OS level // Scientific and Technical Bulletin of the Volga Region. 2021. №12. pp. 149…151.

10. *Ermakov A.S.* Prospective development of DevOps methodology // Bulletin of NSUEM. 2020. №4. pp. 174…183.

11. *Lukashenko M.A., Telegina T.V.* Management of educational product creation using the Scrum method // Azimuth of Scientific Research: Economics and Management. 2019. №2. pp. 223...227.

Рекомендована кафедрой автоматики и промышленной электроники РГУ им. А.Н. Косыгина. Поступила 23.05.24.