МОДЕРНИЗАЦИЯ ЛОКАЛЬНОЙ СЕТИ ПРИ ПОМОЩИ КОНТЕЙНИРИЗАЦИИ И МИКРОСЕРВИСНОЙ АРХИТЕКТУРЫ

Сазонов А.А., студент

Белорусский государственный университет информатики и радиоэлектроники,
Институт информационных технологий,
г. Минск, Республика Беларусь

Шелягович А.С.– маг. техн. наук, ассистент каф. ИСиТ

В работе описывается модернизация локальной сети при помощи контейнеризации и микросервисной архитектуры, проанализированы преимущества и вызовы такого подхода, проведено сравнение контейнеризации с виртуализацией, а также описаны основные этапы реализации проекта модернизации. Также в работе рассмотрены перспективы развития данного подхода в будущем.

Микросервисная архитектура — это подход к разработке программного обеспечения, в котором приложение разбивается на более мелкие, независимые компоненты, называемые микросервисами. Каждый микросервис выполняет свою конкретную задачу и может быть разработан, развернут и масштабирован независимо от других микросервисов. Монолитная архитектура, напротив, представляет собой единую программную систему, которая разбита на отдельные модули, но все они работают в едином контексте и зависят друг от друга.

Микросервисная архитектура имеет следующие преимущества по сравнению с монолитной: гибкость и масштабируемость - микросервисы могут быть разработаны, развернуты и масштабированы независимо друг от друга, что позволяет быстро адаптироваться к изменениям в бизнес-требованиях и увеличивать масштаб приложения при необходимости; улучшенная отказоустойчивость - если один микросервис не работает, это не повлияет на работу других микросервисов, что позволяет быстро выявлять и устранять проблемы; ускоренная разработка - каждый микросервис может быть разработан и развернут независимо от других, что позволяет быстро внедрять новые функции и ускорять процесс разработки; лучшая поддержка - микросервисы могут быть написаны на разных языках программирования и использовать разные технологии, что позволяет выбрать наиболее подходящие для каждого компонента; улучшенная безопасность - микросервисы могут быть защищены независимо друг от друга, что повышает безопасность приложения в целом. Пример схемы микросервисной и монолитной архитектуры представлен на рисунке 1. [1]

Конечно, микросервисная архитектура не является универсальным решением для всех проектов, и может быть излишне сложной для небольших приложений. Однако, в больших и комплексных проектах, микросервисная архитектура может значительно улучшить гибкость, масштабируемость и отказоустойчивость приложения.

Контейнеризация — это технология развертывания и управления приложениями, которая позволяет упаковать приложение и все его зависимости в изолированный контейнер, который может быть запущен на любой совместимой с контейнером платформе. Контейнер — это единица программного обеспечения, которая включает в себя приложение и все его зависимости, такие как библиотеки, файлы конфигурации и т.д. Таким образом, контейнер предоставляет изолированную

среду для запуска приложения, что упрощает процесс развертывания и управления приложением в различных средах.

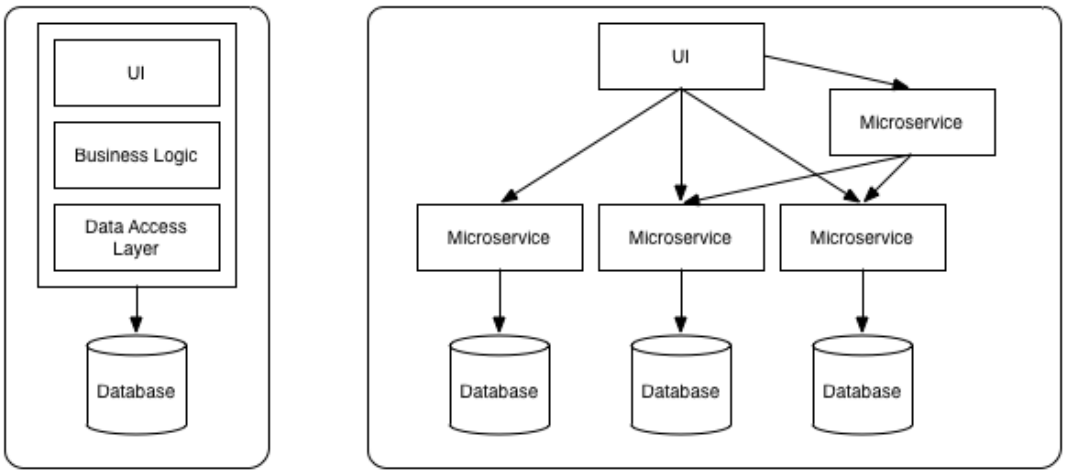

Рисунок 1 – Микросервисная и монолитная архитектура

Основными преимуществами контейнеризации являются портативность, масштабирование, отказоустойчивость и гибкость. Под портативностью понимается возможность развертывания приложений в нескольких типах операционных систем, а также выпуска обновлений до современных версий. Масштабирование позволяет использовать несколько контейнеров для нескольких программ на одном персональном компьютере, сильно не нагружая систему. Отказоустойчивость достигается тем, что один неисправный контейнер не влияет на другие, т.к. они используют изолированные пространства. Гибкость заключается в том, что разработчики могут чинить неисправности и изменять код без вмешательства в работу оборудования, операционных систем и других приложений,

Контейнеризация является одним из ключевых компонентов микросервисной архитектуры, которая позволяет разбить приложение на небольшие, изолированные сервисы, которые легко масштабировать и управлять. Контейнеризация также позволяет повысить безопасность и надежность приложения, так как каждый контейнер работает в изолированной среде и не влияет на другие контейнеры или систему в целом. Пример организации контейнерного подхода с использованием Docker реализован на рисунке 2.

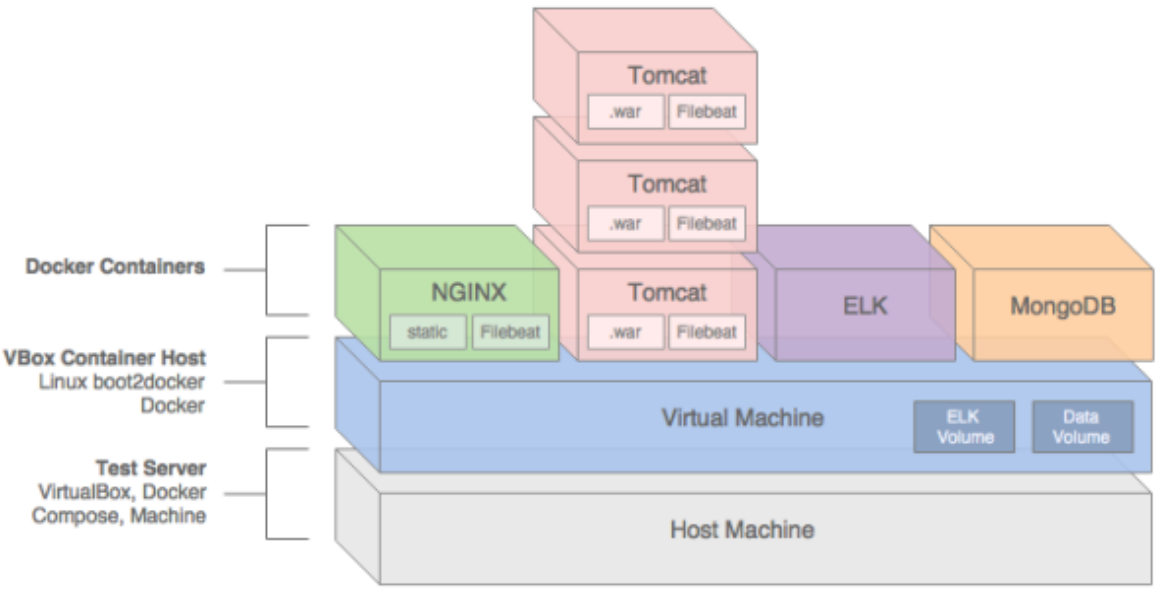

Рисунок 2 – Контейнеры в производственной среде

Контейнеризация и виртуализация имеют разные подходы к хост-машине. Контейнеризация работает на уровне операционной системы и позволяет разделять ресурсы системы между различными приложениями, пакетами и программами, которые работают в изолированных средах.

Виртуализация, с другой стороны, использует гипервизор, который позволяет создавать виртуальные машины для запуска операционных систем. Это означает, что каждая виртуальная машина имеет свой собственный набор ресурсов, включая процессор, память и диски.

В целом, контейнеризация более легковесна и гибка, позволяя запускать более быстрые и эффективные приложения. Виртуализация более надежна и безопасна, но может быть менее гибкой и требует больших ресурсов. [2]

Основные этапы модернизации локальной сети: разбиение монолитного приложения на микросервисы; использование контейнеров (Для развертывания и управления контейнерами можно использовать такие инструменты как Docker, Kubernetes, OpenShift и т.д.); использование инструментов для автоматического развертывания: (Для упрощения процесса развертывания и управления сервисами можно использовать такие инструменты, как Ansible, Chef, Puppet и т.д.); Организация мониторинга и управления: (Для мониторинга и управления сервисами можно использовать такие инструменты, как Prometheus,Zabbix, Grafana, ELK-стек и т.д.); Применение DevOps-практик(CI/CD,IaC,SaC и т.д.).

Микросервисный подход к разработке программного обеспечения получил широкое распространение в последние годы и показал себя как эффективный и гибкий способ создания приложений. Существует несколько факторов, которые говорят о том, что этот подход будет активно развиваться и использоваться в будущем: рост числа устройств и быстродействия сетей, ориентация на контейнеризацию и облака, повышение требований к безопасности, рост требований к персонализации пользовательского опыта.

Исходя из этих факторов, можно сделать вывод о том, что микросервисный подход будет востребован и активно развиваться в будущем.

Список использованных источников:

1. Ньюмен С. От монолита к микросервисам / С. Ньюмен - Издательство «БХВ», 2021. – 272с
2. Руководство по DevOps / Д. Хамбл [и др.] ; под общ. ред. Д. Хамбл - Издательство Манн, 2018. - 507с