Министерство Науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский Томский политехнический Университет»



Инженерная школа информационных технологий и робототехники

Отделение информационных технологий

Направления подготовки: 09.04.01 «Информатика и вычислительная техника»

**Лабораторная работа №3**

**Проектирование и разработка приложений с использованием технологии контейнеризации**

по дисциплине:

**Облачные технологии**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Исполнитель:** |  | | | | |
| студент группы | 8ВМ22 |  | Ямкин Н.Н. |  |  |
|  |  |  |  |  |  |
| **Руководитель:** | . | | | | |
| к.т.н доцент ОИТ, ИШИТР |  |  | Ботыгин И.А. |  |  |
|  |  |  |  |  |  |

Томск – 2023

**Теоретическая часть**

Контейнер приложения — экземпляр исполняемого программного обеспечения (ПО), который объединяет двоичный код приложения вместе со всеми связанными файлами конфигурации, библиотеками, зависимостями и средой выполнения.

Контейнеры — небольшие, быстрые и портативные, потому что могут не включать в себя гостевую операционную систему (ОС), а используют функции и ресурсы основной ОС. Контейнеры используют не только для изоляции различных программных процессов, но и для контроля ресурсов, за которые эти процессы могут конкурировать. Это, например, объем памяти или ресурсы процессора.

Например, на сервере с Ubuntu Linux запущены контейнеры с приложениями, которым требуется Alpine Linux, а также другие контейнеры с приложениями, которым необходима определенная версия Debian.

Чтобы лучше понять, чем хороши контейнеры, стоит сравнить их с виртуальными машинами.

Виртуальные машины — абстракция на уровне физического оборудования, превращает один сервер в несколько. На каждой виртуальной машине (ВМ) отдельная гостевая операционная система работает поверх операционной системы хоста с виртуализированным доступом к базовому оборудованию. Виртуальные машины с разными ОС могут работать на одном физическом сервере: ВМ UNIX может работать вместе с ВМ Linux и так далее. Микроядро и система виртуализации, которые создают и запускают виртуальные машины, называются гипервизорами или мониторами ВМ. Это то, что находится между оборудованием и ВМ и необходимо для виртуализации сервера, а также для изоляции операционных систем друг от друга.

Контейнеры — абстракция на уровне приложения, объединяет код и зависимости. Контейнеры устанавливаются поверх физического сервера и его ОС, например Linux или Windows. Каждый контейнер отделяет свое содержимое от операционной системы. Контейнеры «легкие» — весят всего мегабайты и запускаются за секунды, ведь они берут лишь небольшую часть памяти при совместном использовании ОС.

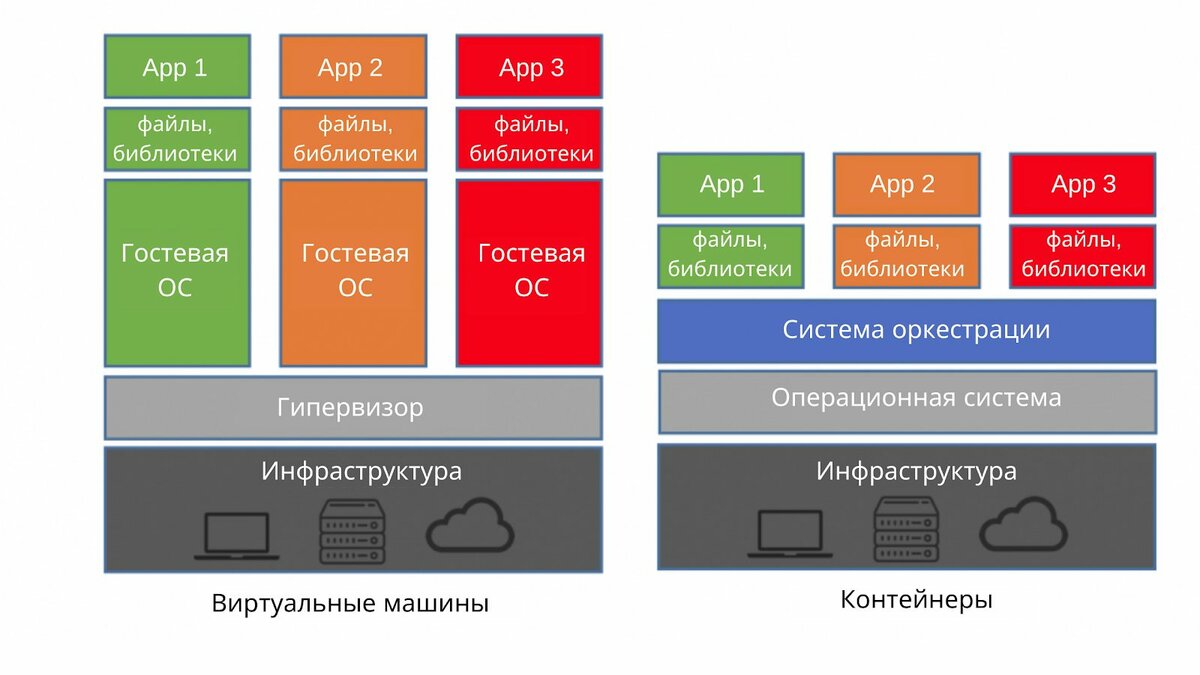


Рисунок 1 – Архитектура

Виртуальные машины могут запускать любое ядро операционной системы независимо от основной операционной системы, контейнер должен быть совместим с ядром ОС сервера.

Как и ВМ, контейнеры позволяют упаковывать приложение вместе с библиотеками и другими зависимостями, обеспечивая изолированные среды для запуска программных сервисов. Они отделяют приложения друг от друга. Это означает, что не придется беспокоиться о конфликтующих зависимостях или конфликте ресурсов, ведь можно установить лимиты ресурсов для каждого контейнера. Важно отметить, что это дополнительный уровень безопасности, поскольку приложения не работают в операционной системе сервера.

**Преимущества контейнеризации приложений**

Контейнерная технология дает ряд преимуществ по сравнению с обычной виртуализацией серверов.

1. Решается проблема зависимостей в разных окружениях. Отлаженное на одном компьютере приложение можно легко развернуть на другом, ведь контейнер содержит все необходимые зависимости. В этом случае говорят о переносимости, гибкости контейнеров.
2. Появляется возможность использовать микросервисные архитектуры. Контейнеры хорошо подходят для приложений на основе микросервисов: можно проверить работоспособность каждого контейнера, ограничить каждую службу определенными ресурсами, запускать и останавливать их независимо друг от друга.
3. Можно заметно сократить время разработки приложения. Некоторые технологии, например кеширование слоев сборки, способствуют ускорению циклов разработки и тестирования.
4. Снижение накладных расходов. Контейнеры совместно используют системное ядро операционной системы сервера, следовательно, запуск контейнера не требует запуска отдельного экземпляра ОС для каждого приложения. Это повышает эффективность сервера и снижает затраты на сервер и лицензирование.
5. Легковесность и портативность благодаря тому, что каждый контейнер не содержит образ ОС.
6. Эффективность. Контейнеры позволяют быстрее развертывать приложения, легче масштабировать их горизонтально, проще находить в них ошибки.
7. Изоляция ошибок. Выход из строя одного контейнера не влияет на дальнейшую работу других контейнеров.

**Плюсы и минусы контейнеризации**

Минус: если для работы приложения требуется другая ОС, а не та, что установлена на сервере.

Плюсы:

* для упрощения процесса развертывания и сопровождения приложений;
* для запуска небезопасного или непроверенного кода с целью тестирования или отладки — для этого контейнеры подходят в 99% случаев;
* для запуска приложений, требующих другого дистрибутива ОС (системные контейнеры);
* для передачи отдельных компонентов приложения между членами команды в ходе цикла «разработка — тестирование — внедрение» и быстрого внесения изменений;
* для микросервисов, которые можно разрабатывать и обновлять независимо;
* для горизонтально масштабируемых приложений — когда запускается несколько одинаковых контейнеров на текущих ресурсах без увеличения стоимости этих ресурсов. В отличие от вертикального масштабирования, где увеличение количества ядер CPU, объемов RAM, размера HDD на сервере стоит денег;
* для модернизации и миграции существующих приложений в более современные среды.

**Практическая часть**

Установим Docker и сопутствующие пакеты на созданную ранее виртуальную машину в Yandex Cloud с операционной системой Ubuntu Lunix 22.04.

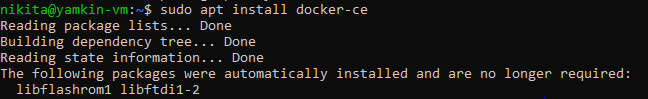


Рисунок 2 – Виртуальная машина

Далее установим Docker.

|  |
| --- |
| nikita@yamkin-vm:~$ sudo apt update  sudo apt install apt-transport-https ca-certificates curl software-properties-common  curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add –  sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"  sudo apt install docker-ce  sudo systemctl status docker |

Docker установлен.

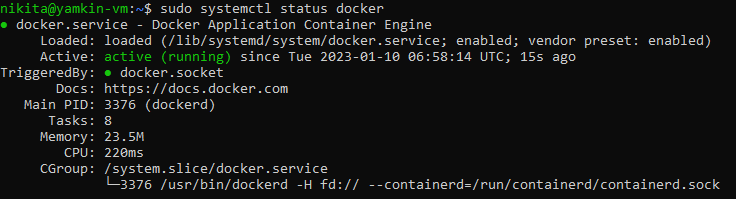


Рисунок 3 – Docker установлен



Рисунок 4 – Docker установлен: версия

Docker ps показывает, какие контейнеры активны сейчас.



Рисунок 5 – Docker активные контейнеры

Docker images показывает список images.

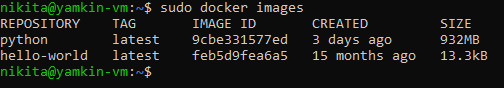


Рисунок 6 – Docker Images

Тестовая проверка работы docker.

|  |
| --- |
| sudo docker run hello-world |

Произойдет поиск image с именем hello-world, если его нет, то он скачает его из интернета.

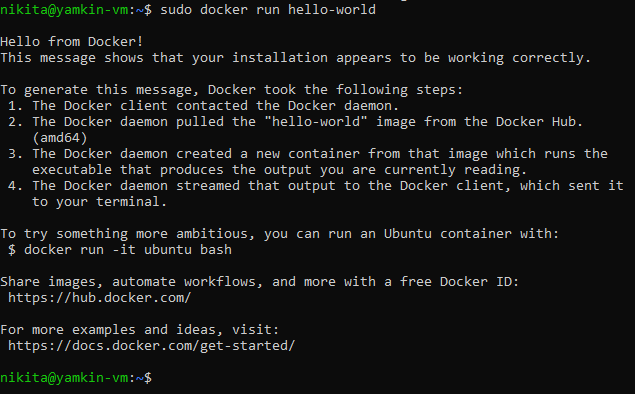


Рисунок 7 – Docker Images hello-world

Image скачан.

Далее попробуем скачать докер файл для запуска Python. Вообще все готовые image находятся в публичном репозитории Docker Hub <https://hub.docker.com/search?type=image>, в данном репозитории можно найти image, который может вам пригодиться и не надо тратить время на его создание с нуля.

Посмотрим, какие есть images для запуска Python в Docker Hub.

|  |
| --- |
| nikita@yamkin-vm:~$ sudo docker search python |

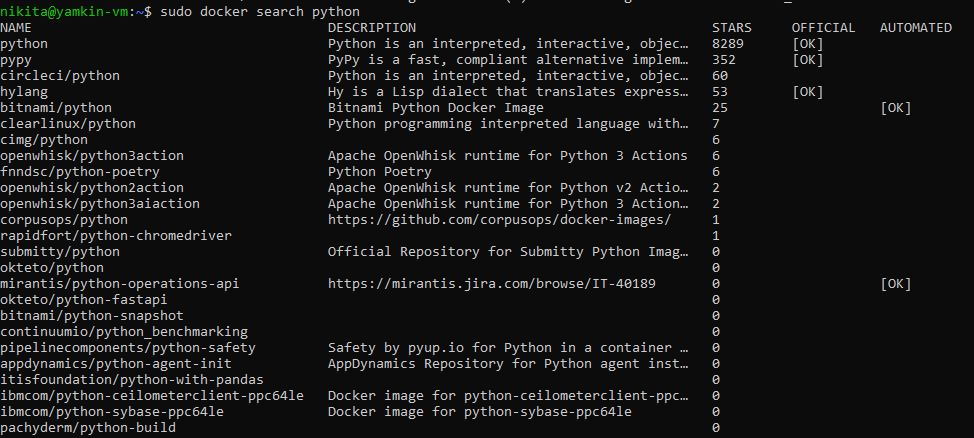


Рисунок 8 – Результаты поиска

Скачаем image от официального дистрибьютора и запустим его.

|  |
| --- |
| nikita@yamkin-vm:~$ sudo docker pull python  nikita@yamkin-vm:~$ sudo docker run -it python |

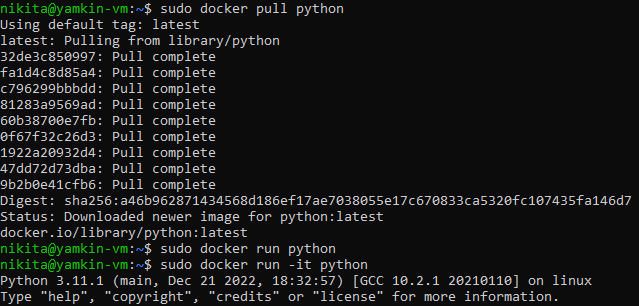


Рисунок 9 – Запуск Python

Напишем несколько простых команд в консоли Python.

|  |
| --- |
| >>> print('Hello from Nikita!')  Hello from Nikita!  >>> print(1+3)  4 |

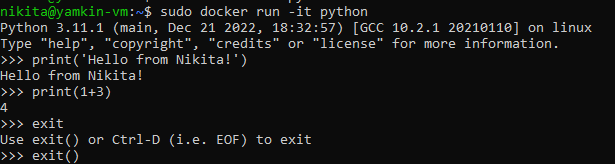


Рисунок 10 – Написание простейших команд в Python

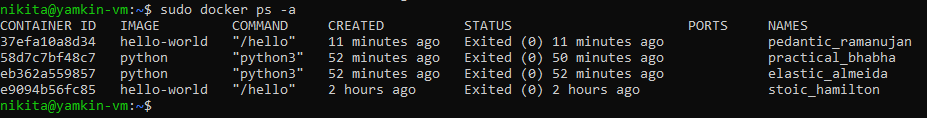


Рисунок 11 – Список активных контейнеров

**Заключение**

В ходе лабораторной работы была проведена ознакомительная работа с контейнерами и контейнеризацией приложений.

Так в процессе работы на виртуальной машине Ubuntu Lunix 22.04 был запущен Python при помощи Docker.

**Список использованных источников**

1. Основы виртуализации. Лабораторная работа №3 // YouTube.com. URL: <https://www.youtube.com/watch?v=JWTj7WDZpfw> (дата обращения: 10.01.2023).
2. Виртуализация и контейнеризация / Docker // YouTube.com. URL: <https://www.youtube.com/watch?v=h6LNP2Amp4M> (дата обращения: 10.01.2023).