**Конспект.**

**Docker** — это ПО для автоматизации развертывания и управления приложениями в средах с поддержкой контейнеризации. Он помогает «упаковать» приложение со всем его окружением и зависимостями в образ, который достаточно просто разворачивается в контейнере в любой Linux системе.

**Контейнер** — это изолированный процесс, в который вы упаковываете свое приложение со всеми зависимостями.

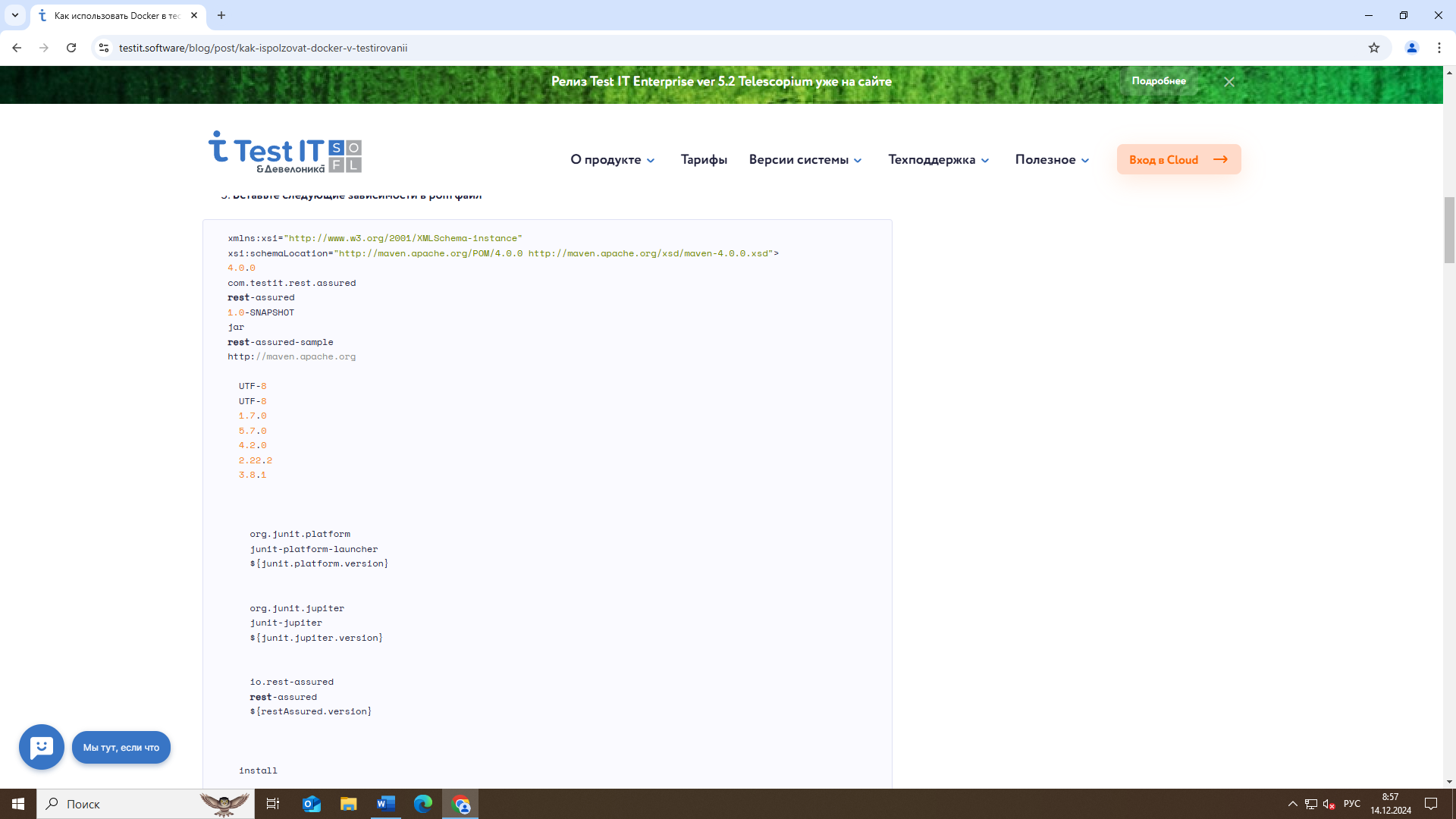
**Почему стоит использовать docker в тестах**

1. Проблема запуска тестов в команде без установки стека тестовых технологий (solenoid, браузеры и др.)
2. Менее требовательный, чем виртуальная машина по ресурсам (масштабирование ограничивается только ресурсами памяти и процессора на узлах с docker)
3. Проблема с инсталляцией дополнительного софта в разных контурах т.к. имеются ограничения
4. Использование TestContainers решает многие проблемы. Например, запуск kafka в docker для работы приложения или интеграционных сервисов.

**Главными преимуществами** докера является его простота и дружелюбность в поведении в инструментах автоматизации. Таким образом, с помощью docker вы всегда сможете подтянуть нужные вам образа с программным обеспечением из docker hub либо собрать образ самостоятельно (проблема с заявками по инсталляции доп. ПО на ваш сервер уходит в прошлое). Например, вы можете использовать готовые docker image: kafka, PostgreSQL, Selenoid и др. Также можно отметить, что запуск тестов можно выполнить с любого удобного вам сервера вместо агента. Данная технология кроссплатформенная, т.е. запускается в разных ОС.

**К минусам** можно отнести повышенной порог входа для работы, а также особенности масштабирования.

**Сборка docker image, запуск docker контейнера, запуск тестов в контейнере**

1. Установите docker desktop
2. Создайте проект с пакетным менеджером maven в intellij idea
3. Вставьте следующие зависимости в pom файл
4. Создайте структуру папок

├───src

│ └───test

│ ├───java

│ │ └───com

│ │ ├───apiTests

│ │ ├───model

│ └───resources

1. Создайте модели (DTO)

Создадим класс UserDTO с моделью для выполнения запросов GET, POST, PUT, DELETE в каталоге src/test/java/com/model:

**public** **class** **UserDTO** {

**public** String name;

**public** String job;

**public** **UserDTO**(String name, String job) {

**this**.name = name;

**this**.job = job;

    }}

1. Напишем простой автотест для сайта[https://reqres.in](https://reqres.in/) для запросов GET, POST, PUT, DELETE с разным способом валидации Response
2. Создадим DockerFile

**DockerFile** — это инструкция по созданию образа.

1. Выполним сборку docker image:

* Откройте терминал intellij idea в корне проекта
* Выполните команду: docker build -t test-docker:1.0 .

**docker** — запуск команды Docker

**build** — команда сборки образа Docker

**-t** — присвоить образу tag с названием 1.0

**test-docker** — имя образа

**. [точка]** — указывает, что DockerFile находится в текущей папке

* Проверьте Docker Desktop. В разделе images появился новый образ test-docker c tag 1.0 или введите в терминале команду: *docker images*

**docker** — запуск команды Docker

**images** — запуск команды вывода идентификаторов образов

1. Выполните запуск тестов следующим образом:

* docker run --name teststartdocker -p 4444:4444 test-docker:1.0

**docker** — запуск команды Docker

**run** — команда запуска образа Docker

**--name run-test-docker** — присваивание имя контейнера

**-p 4444:4444** — порты на входе и выходе контейнера

**test-docker:1.0** — имя образа и версия(tag) для запуска контейнера

1. Остановка docker контейнера выполняется следующим образом docker stop [NAME или IMAGE+TAG]

**Основные команды docker:**

1. **docker ps —** просмотр списка запущенных контейнеров
2. **docker pull —** загрузка образа**.**
3. **docker build** — собирает образ.
4. **docker logs**— смотрим логи
5. **docker run** — запускаем контейнер на основе указанного образа.
6. **docker stop**— останавливает контейнер.
7. **docker kill**— «убивает» контейнер.
8. **docker rm** — удаляет контейнер.
9. **docker rmi**— удаляет образ.
10. **docker volume ls** — список томов. Данная команда показывает список томов, которые являются основным механизмом для хранения данных, генерируемых контейнерами Docker.