Департамент образования и науки города Москвы

Государственное автономное образовательное учреждение высшего образования города Москвы

«Московский городской педагогический университет»

Институт цифрового образования

Департамент информатики, управления и технологий

ДИСЦИПЛИНА:

Интеграция и развертывание программного обеспечения с помощью контейнеров  
**Лабораторная работа №1**

**Тема:**

«Установка и настройка Docker. Работа с контейнерами в Docker»

Выполнил(а): Сачкова Г.Г, группа: АДЭУ-211

Преподаватель: Босенко Т.М.

Москва

2025

**Вариант 11. Задача:** загрузить образ jenkins, запустить контейнер, настроить маршрутизацию портов и выполнить начальную настройку Jenkins.

**Цель работы:** освоить процесс установки и настройки Docker, научиться работать с контейнерами и образами Docker.

**Ход работы:**

1. Необходимо убедиться, что docker установлен и готов к работе. Для этого выполняем ряд команд и проверяем, что все ответы успешны. Результат выполнения приведен на рисунке 1.

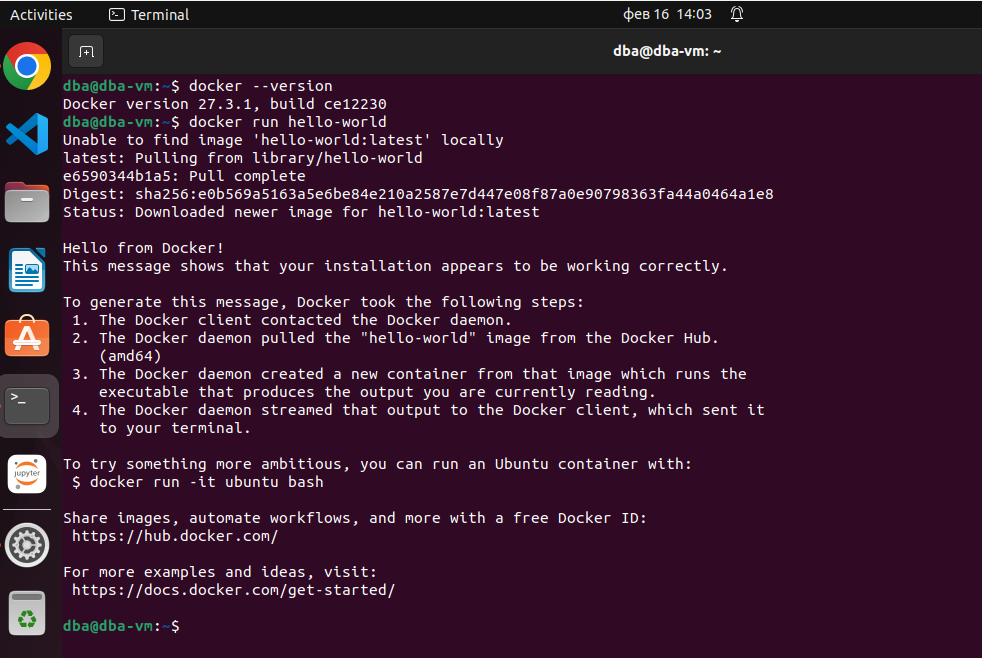


Рисунок 1 – работоспособность docker

1. Для знакомства с основными командами Docker CLI был выполнен ряд команд: для просмотра локальных образов, списка запущенных команд, а также списка всех контейнеров. Выполнение данных команд отображено на рисунке 2.

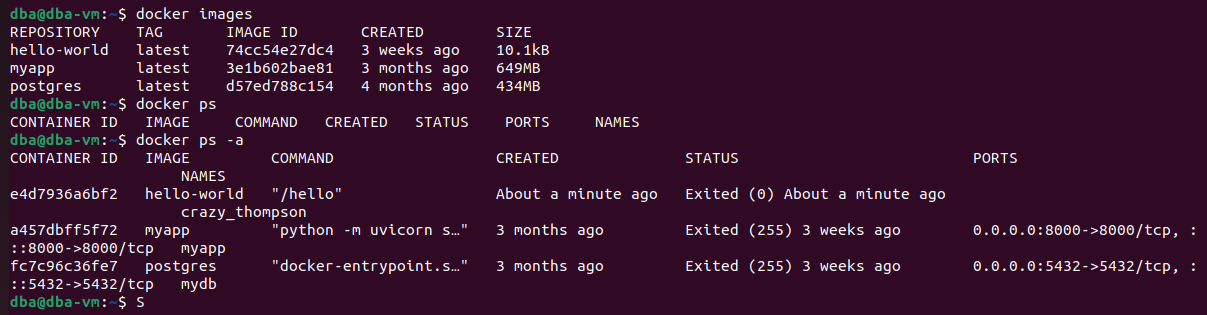


Рисунок 2 – Основные команды Docker CLI.

1. Далее переходи к выполнению индивидуального задания. Первым делом мы загружаем образ Jenkins. Команда и результат загрузки представлены на рисунке 3.

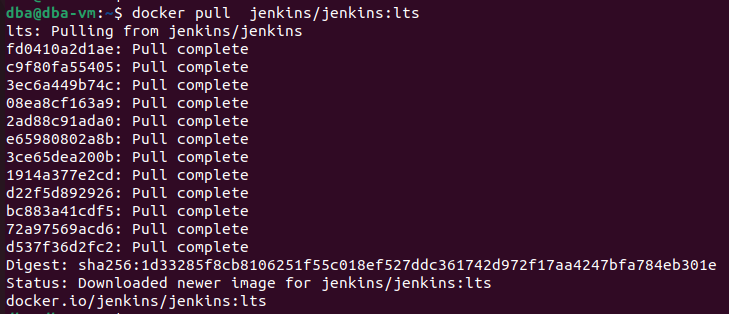


Рисунок 3 – Загрузка образа Jenkins

1. Запускаем контейнер Jenkins с маршрутизацией портов 8080 и 50000. Сразу проверяем, что контейнер запущен. Вывод на рисунке 4.

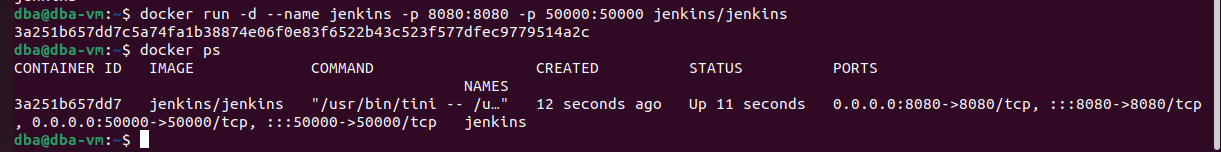


Рисунок 4 – Запуск контейнера

1. При переходе по адресу <http://localhost:8080> мы видим страницу для входа Jenkins. Об этом свидетельствует рисунок 5

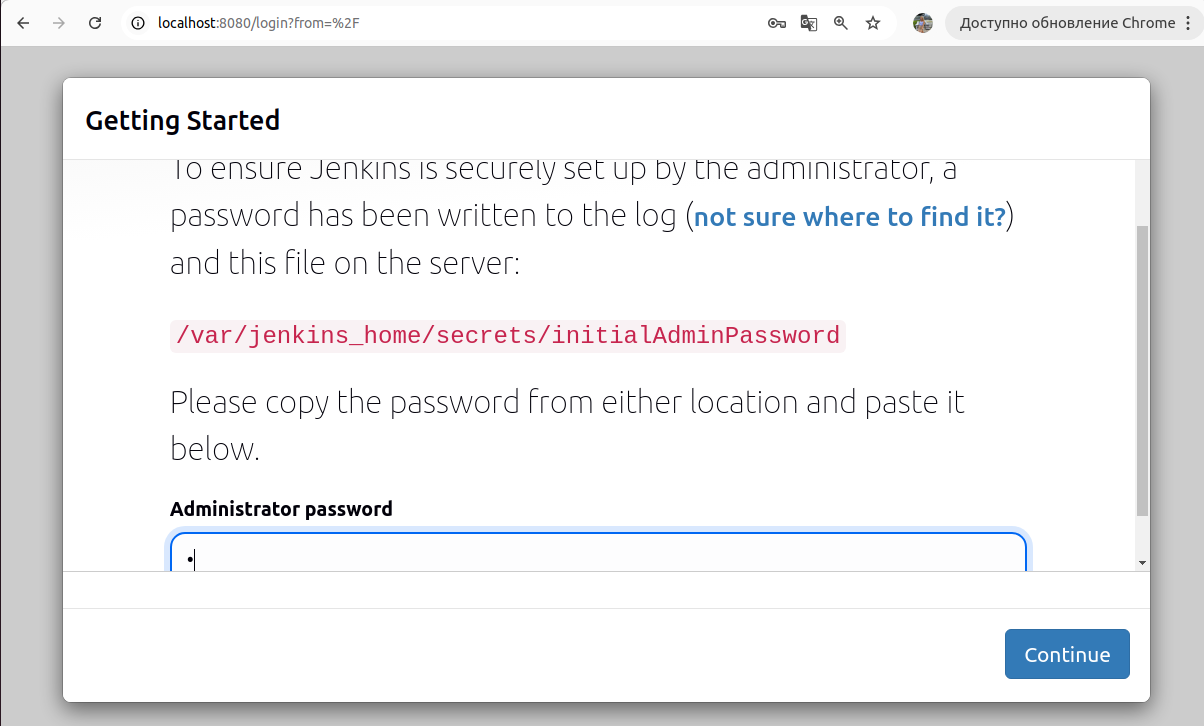


Рисунок 5 – Jenkins

1. Для проведения начальной настройки Jenkins был получен пароль (рисунок 6). И далее была произведена первоначальная настройка: установлены стандартные пакеты (рисунок 7), создан администратор (рисунок 8). После проведения настройки, можно ознакомиться с интерфейсом и функционалом (рисунок 9).



Рисунок 6 – Получение пароля для входа

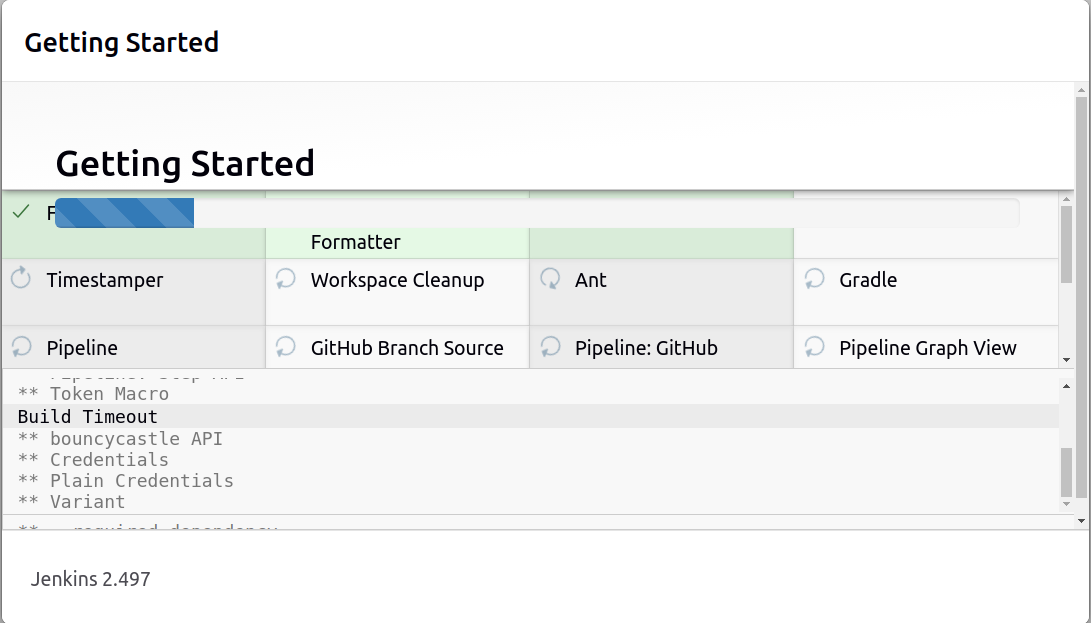


Рисунок 7 – Загрузка пакетов

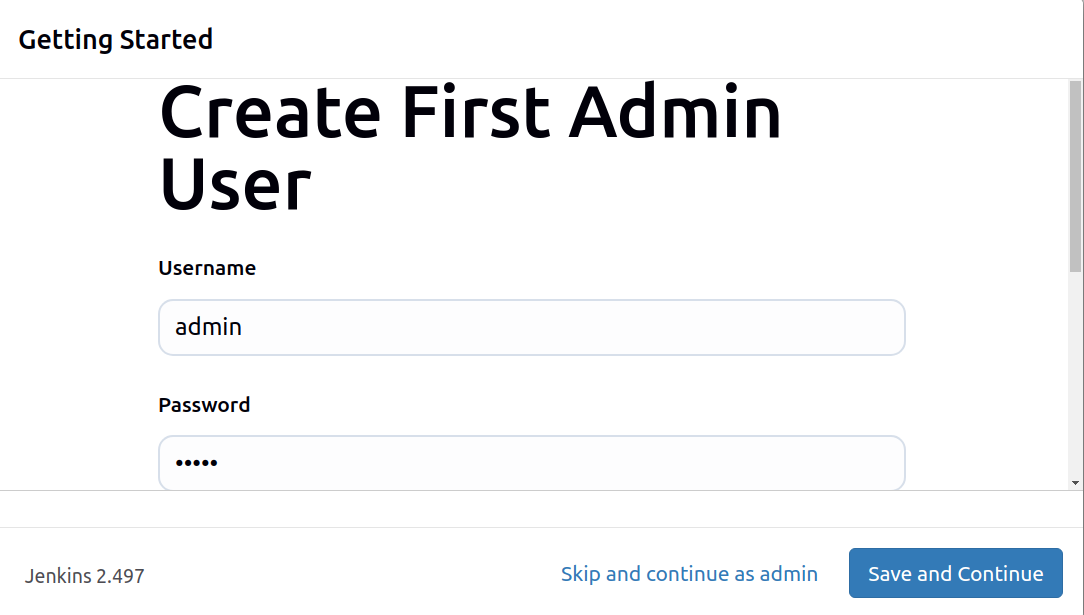


Рисунок 8 – Создание администратора

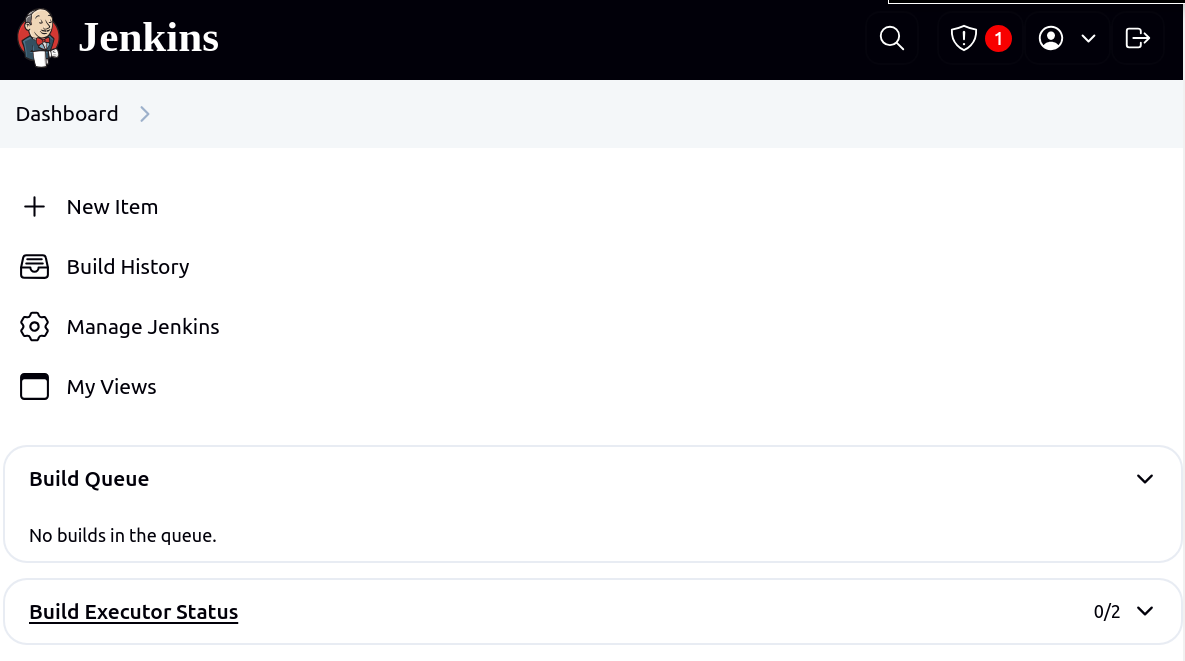


Рисунок 9 – Интерфейс Jenkins

1. После завершения работы с Jenkins контейнер был остановлен и удален (рисунок 10)

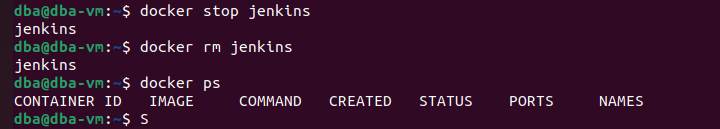


Рисунок 10 – Остановка и удаление контейнера

**Заключение**

Jenkins — программная система с открытым исходным кодом на Java, предназначенная для обеспечения процесса непрерывной интеграции программного обеспечения.

Задачи Jenkins:

* Позволяет автоматизировать часть процесса разработки программного обеспечения, в котором не обязательно участие человека
* Обеспечивает непрерывную интеграцию (Continuous Integration, CI) — один из этапов разработки, на котором происходит сборка рабочих копий проекта в единый макет-черновик, их тестирование, доставка или развёртывание программного обеспечения.
* Во время интеграции можно выявить слабые места и возможные ошибки в проекте и сразу их исправить.
* Оптимизирует рабочий процесс: не нужно нанимать штат профессиональных программистов, в Jenkins можно разобраться даже без специальной подготовки.

**Контрольные вопросы**

1. Что такое Docker и для чего он используется?

Docker — это платформа для разработки, доставки и запуска приложений в контейнерах. Контейнеры позволяют упаковать приложение со всеми его зависимостями (библиотеками, настройками, средой выполнения) в изолированную среду, которая может работать на любой системе, поддерживающей Docker. Это обеспечивает переносимость и согласованность работы приложения на разных этапах разработки и развертывания.

Docker используется для:

* Упрощения разработки и тестирования приложений.
* Обеспечения одинаковой среды на всех этапах (разработка, тестирование, production).
* Ускорения развертывания приложений.
* Изоляции приложений и их зависимостей.

1. Какие преимущества дает использование контейнеров Docker по сравнению с виртуальными машинами?

Эффективность использования ресурсов:

Контейнеры используют общее ядро операционной системы хоста, что делает их более легковесными по сравнению с виртуальными машинами, которым требуется отдельная ОС для каждого экземпляра.Контейнеры запускаются быстрее и потребляют меньше памяти и процессорных ресурсов.

Переносимость:

Контейнеры содержат все зависимости приложения, что позволяет запускать их на любой системе с Docker без необходимости настройки среды.

Масштабируемость:

Контейнеры легко масштабируются, что особенно полезно в микросервисных архитектурах.

Изоляция:

Контейнеры изолируют приложения друг от друга, но при этом могут взаимодействовать через сети Docker.

Упрощение управления:

Docker предоставляет удобные инструменты для управления жизненным циклом контейнеров (создание, запуск, остановка, удаление

1. Что такое образ Docker и как он связан с контейнерами?

Образ Docker (Docker Image) — это шаблон, содержащий все необходимые файлы, зависимости и настройки для запуска приложения. Образы создаются на основе Dockerfile, который описывает, как собрать образ.

Образы являются неизменяемыми (immutable). После создания их нельзя изменить, но можно использовать для создания новых образов.

Контейнеры — это запущенные экземпляры образов. Когда запускается контейнер, Docker создает из образа изолированную среду выполнения.Контейнер работает как изолированный процесс на хосте.

1. Какие основные команды Docker CLI вы узнали в ходе выполнения лабораторной работы?

docker ps

docker ps -a

docker images

docker run

docker stop

docker rm

1. Как можно настроить маршрутизацию портов при запуске контейнера Docker?

Маршрутизация портов позволяет перенаправлять порты контейнера на порты хоста. Это делается с помощью флага -p при запуске контейнера