|  |
| --- |
| A black and white emblem with two birds and a scroll  AI-generated content may be incorrect. |
| МИНОБРНАУКИ РОССИИ  Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |

**РАБОТА ДОПУЩЕНА К ЗАЩИТЕ**

Руководитель

программы \_\_\_\_\_\_\_\_\_\_\_ А.Т. Тарланов

«30» мая 2025 г.

**ИТОГОВАЯ АТТЕСТАЦИОННАЯ РАБОТА**

по дополнительной программы профессиональной переподготовки

«Технологии Девопс»

На тему: **«**Набор тематических заданий по проведению итоговой аттестации DevOps-инженера – вариант 287**»**

Обучающийся *Враженко Даниил Олегович*

*Подпись Фамилия, имя, отчество*

|  |  |  |
| --- | --- | --- |
|  |  |  |
| группа | ИКБО-50-23 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Руководитель работы** | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *подпись* |  | А.Т. Тарланов |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Москва 2025 г.

СОДЕРЖАНИЕ

[Введение 3](#__RefHeading___Toc52862_883944595)

[Модуль 6. Администрирование баз данных 4](#__RefHeading___Toc52864_883944595)

[Модуль 8. Мониторинг, логирование и оповещение событий 7](#__RefHeading___Toc1267_3933951894)

[Модуль 9. Виртуализация в DevOps 13](#__RefHeading___Toc1719_1609335944)

[Модуль 12. Конфигурационное управление. Что такое IaC 15](#__RefHeading___Toc10294_3688060152)

[Модуль 13. Системы контроля версий. Распределённая система управления версиями Git 18](#__RefHeading___Toc4772_492110456)

[Модуль 14. Жизненный цикл ПО 21](#__RefHeading___Toc1723_1609335944)

[Модуль 15. Практические навыки работы с Docker 28](#__RefHeading___Toc8976_3330932211)

[Модуль 16. Микросервисы и микросервисная архитектура 30](#__RefHeading___Toc8978_3330932211)

[Заключение 32](#__RefHeading___Toc52866_883944595)

[Список литературы 35](#__RefHeading___Toc52868_883944595)

# ****Введение****

Современная DevOps-экосистема требует комплексного подхода, объединяющего администрирование инфраструктуры, автоматизацию процессов, контроль версий и мониторинг систем. Рост спроса на специалистов, способных работать с распределёнными системами, контейнеризацией, инфраструктурой как код и микросервисными архитектурами, делает освоение ключевых инструментов DevOps критически важным для ИТ-отрасли.

Данная итоговая работа посвящена практическому освоению технологий, формирующих основу DevOps-практик. В рамках выполнения заданий модулей №6, 8, 9, 12, 13, 14, 15 и 16 дополнительной профессиональной программы «Технологии Девопс» были реализованы следующие аспекты:

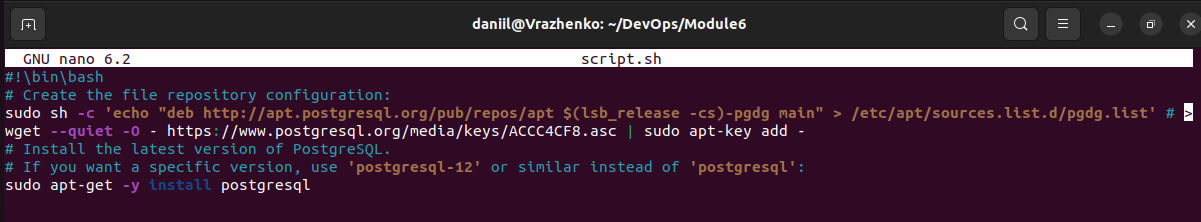
* Администрирование PostgreSQL: от установки СУБД до проектирования структуры данных и управления пользователями;
* Настройка полноценного стека мониторинга на базе Prometheus, Grafana и Node Exporter;
* Конфигурационное управление инфраструктурой через Ansible (Infrastructure as Code);
* Работа с системами контроля версий: развёртывание Git-сервера, branching-стратегии и управление репозиториями;
* Автоматизация жизненного цикла ПО через настройку Jenkins-сервера;
* Освоение инструментов виртуализации и управления инфраструктурой;
* Практическая работа с Docker: создание образов, управление контейнерами и volumes;
* Организация микросервисной архитектуры с использованием Docker Compose.

Практические задания охватывают полный цикл DevOps-процессов — настройку окружения, развёртывание сервисов, оркестрацию контейнеров, конфигурацию инструментов мониторинга и автоматизацию сборки.

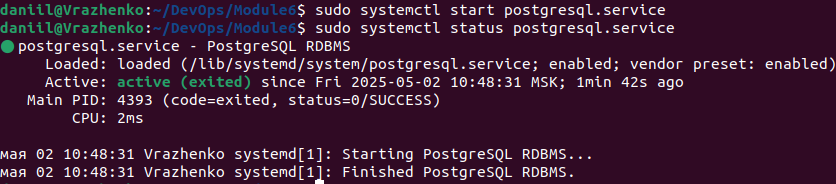
Цель работы — формирование профессиональных компетенций в области DevOps через практическое взаимодействие с ключевыми технологиями: управление базами данных, конфигурационное управление, настройка систем мониторинга, организация CI/CD-процессов, работа с распределёнными системами контроля версий, контейнеризация и построение микросервисных архитектур. Результаты демонстрируют готовность к реализации реальных задач в области сопровождения и развития ИТ-инфраструктуры.

# Модуль 6. Администрирование баз данных

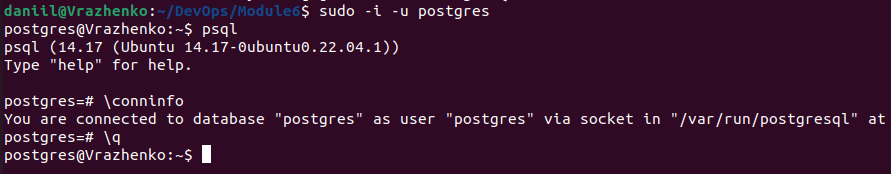
Установим postgresql 12 на ubuntu (рис. 1):

Рисунок 1 - Установка PostgreSQL

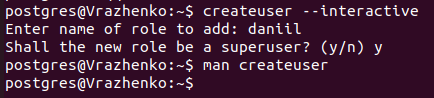
Проверим работоспособность базы данных (рис. 2):

Рисунок 2 - Проверка работоспособности

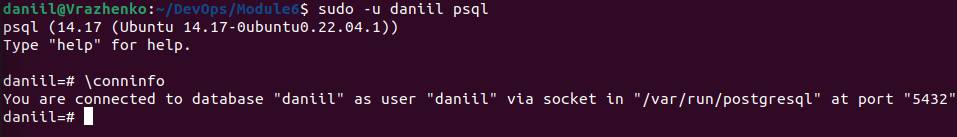
Войдём в potgres (рис. 3):

Рисунок 3 - Вход в postgres

Создадим пользователя (рис. 4):

Рисунок 4 - Создание пользователя

Создадим базу данных (рис. 5):

Рисунок 5 - Создание базы данных

Создадим и просмотрим таблицу в базе данных (рис. 6):

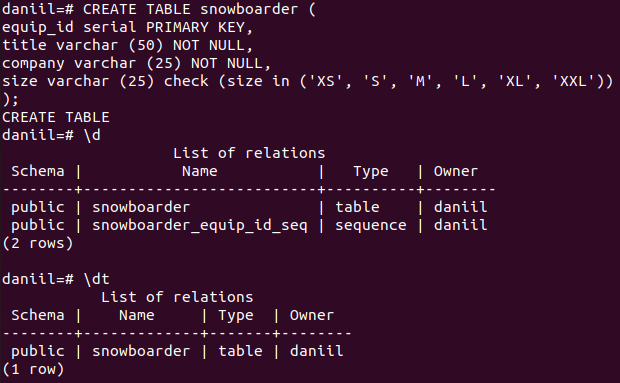
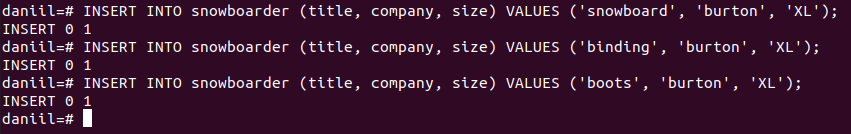
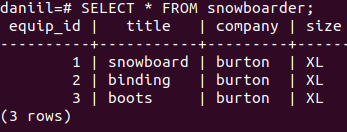


Рисунок 6 - Создание и просмотр базы данных

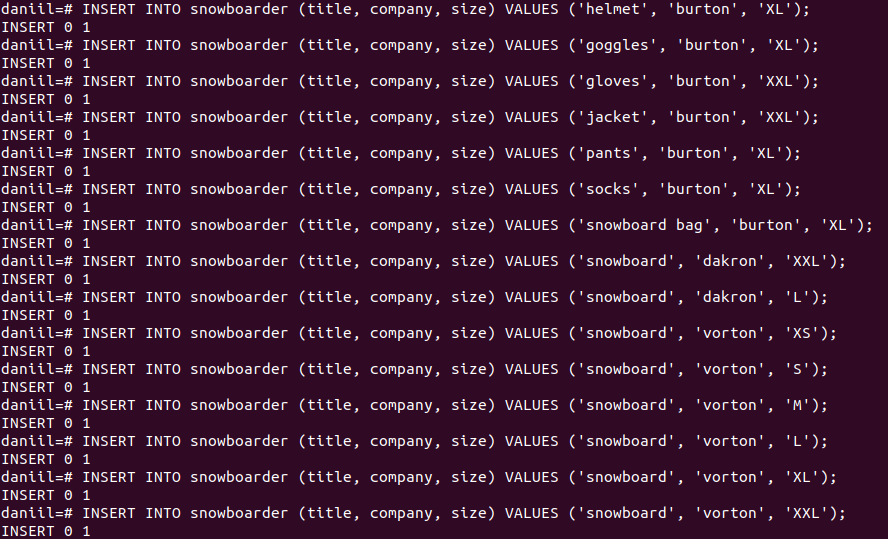
Заполним таблицу данными (рис. 7):

Рисунок 7 - Заполнение таблицы данными

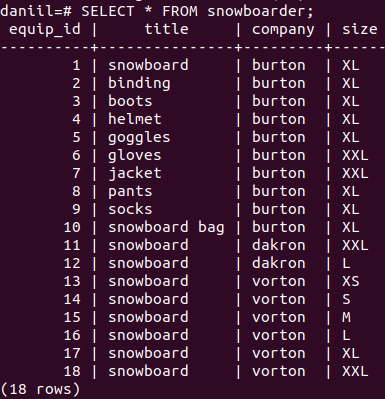
Проверим таблицу (рис. 8):

Рисунок 8 - Проверка таблицы

Дополним таблицу новыми данными (рис. 9):

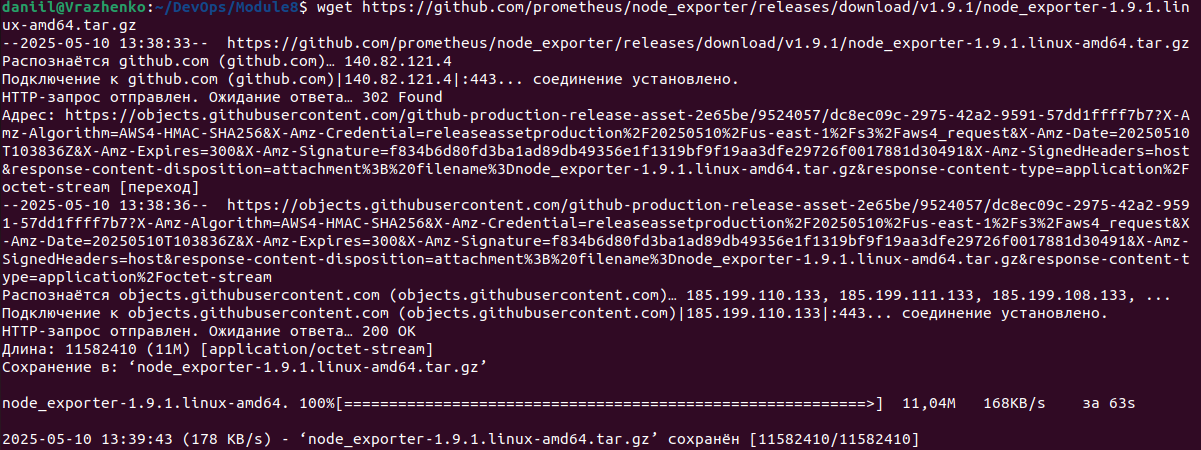
Рисунок 9 - Дополнение таблицы данными

Снова проверим таблицу (рис. 10):

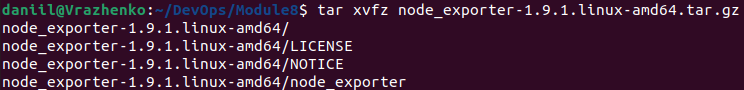
Рисунок 10 - Повторная проверка таблицы

# Модуль 8. Мониторинг, логирование и оповещение событий

Загрузим сжатый пакет node\_exporter (рис. 11):

Рисунок 11 - Загрузка node\_exporter

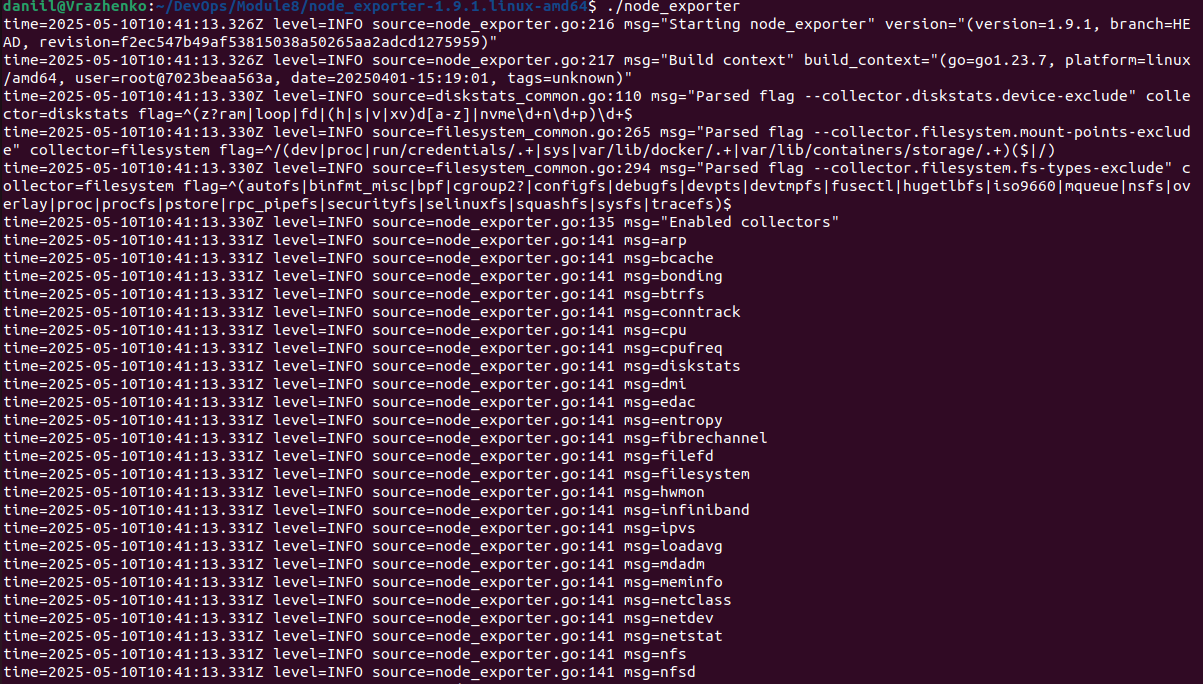
Извлечем двоичный файл node\_exporter (рис. 12):

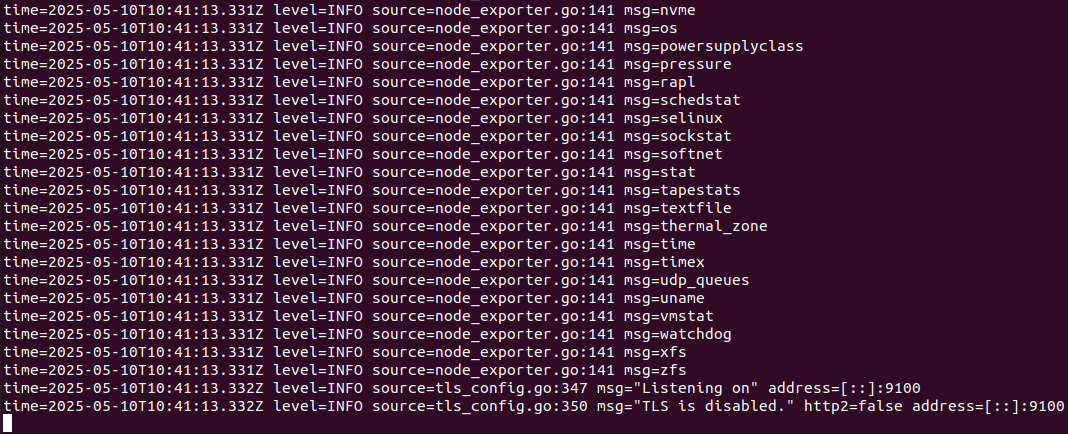
Рисунок 12 - Извлечение двоичного файла node\_exporter

Перейдем в каталог и сделаем двоичный файл исполняемым (рис. 13):

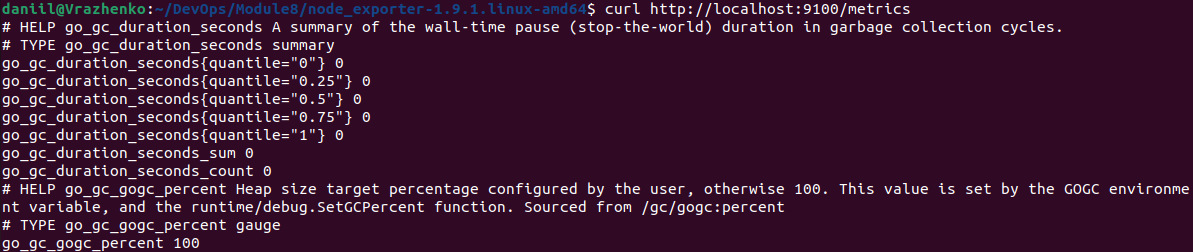
Рисунок 13 - Переход в каталог и изменение прав доступа

Запустим node\_exporter и увидим вывод, указывающий на то, что Node Exporter теперь запущен на порту 9100 (рис. 14-15)

Рисунок 14 - Первая часть вывода

Рисунок 15 - Вторая часть вывода

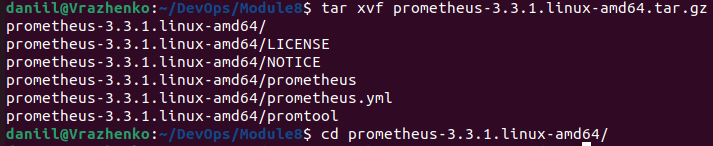
Проверим, что метрики экспортируются через порт 9100 (рис. 16):

Рисунок 16 - Наименьшая часть вывода

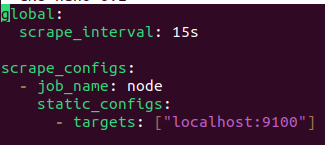
Загрузим архив пакета Prometheus (рис. 17):

Рисунок 17 - Загрузка пакета Prometheus

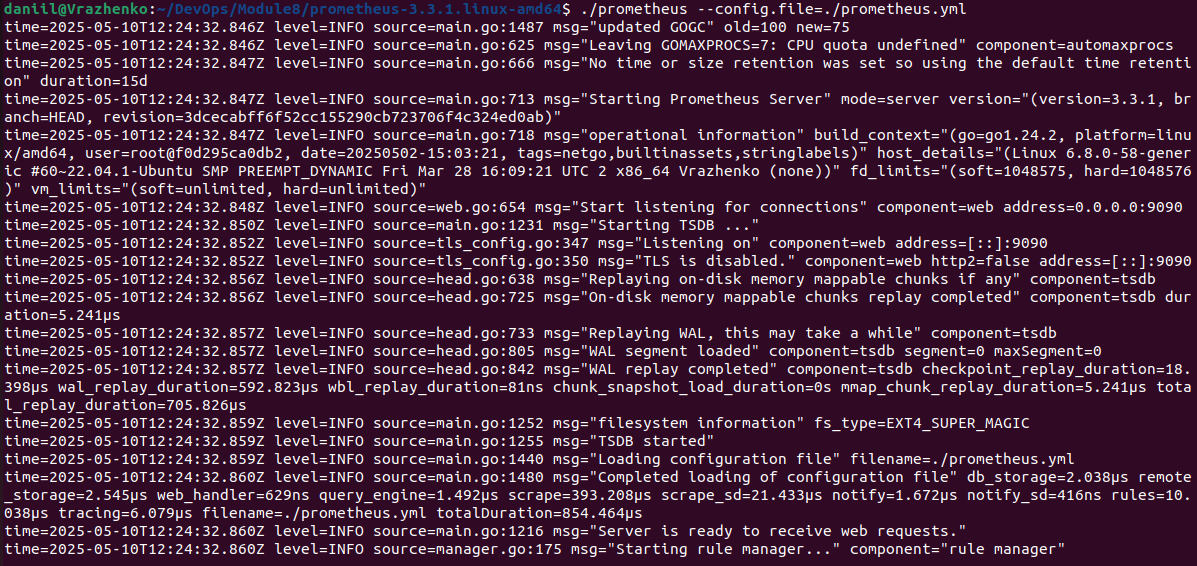
Извлечем двоичный файл и перейдем в каталог (рис. 18):

Рисунок 18 - Извлечение двоичного файла и переход в каталог

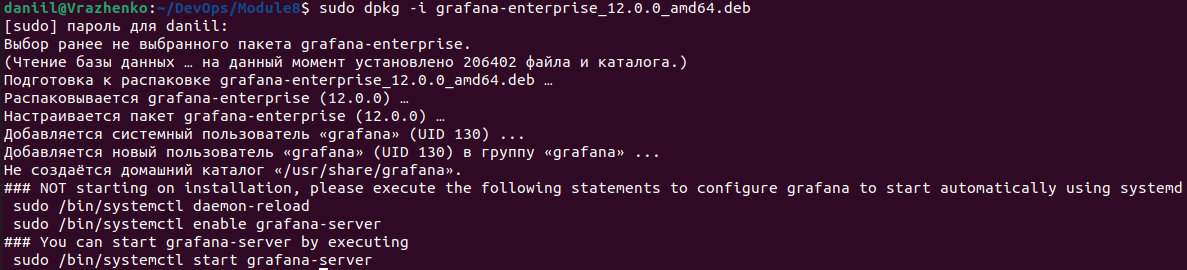
Настроим Prometheus с помощью изменения содержимого файла prometheus.yml (рис. 19):

Рисунок 19 - Содержимое prometheus.yml

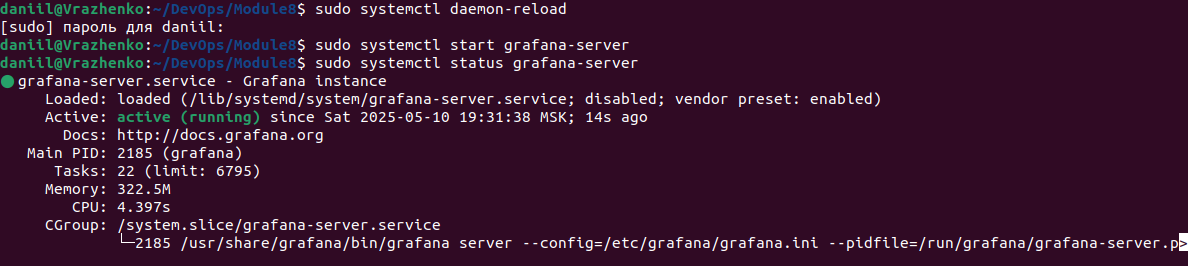
Запустим Prometheus (рис. 20):

Рисунок 20 - Запуск Prometheus

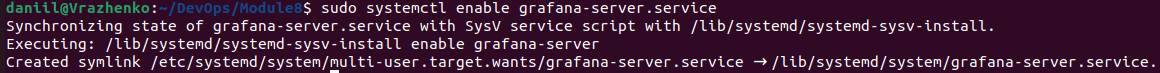
Выполним установку с помощью пакета .deb (рис. 21):

Рисунок 21 - Распаковка пакета .deb

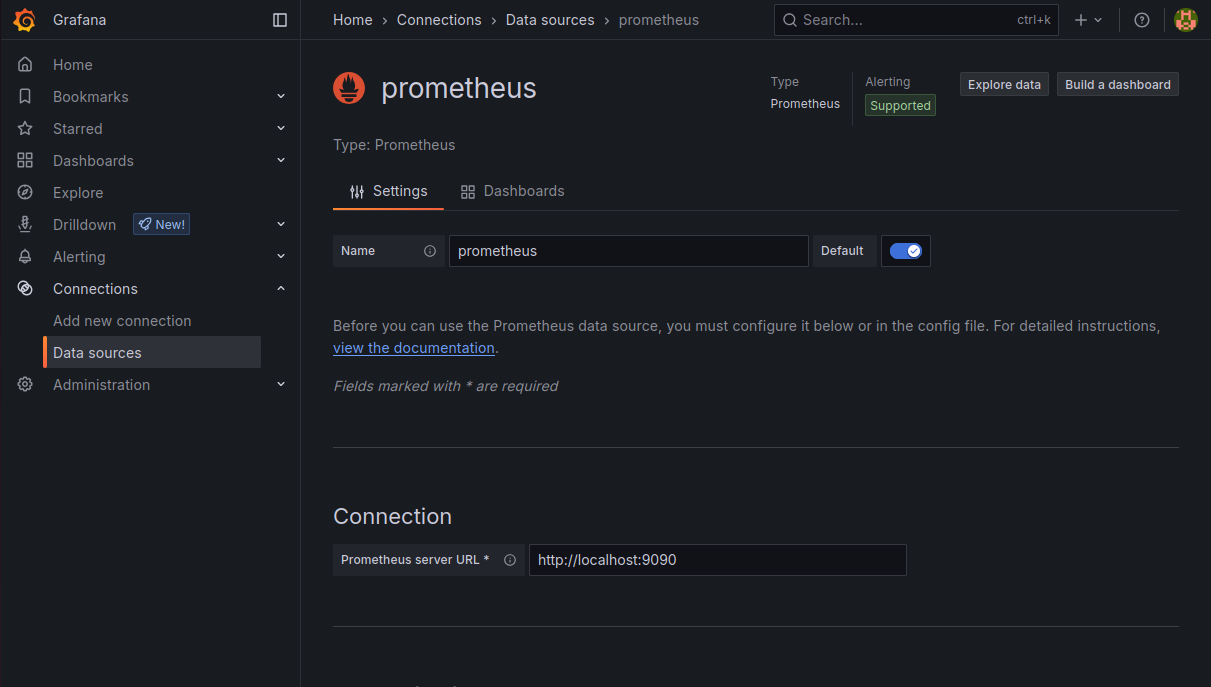
Запустим grafana-server (рис. 22):

Рисунок 22 - Запуск grafana-server

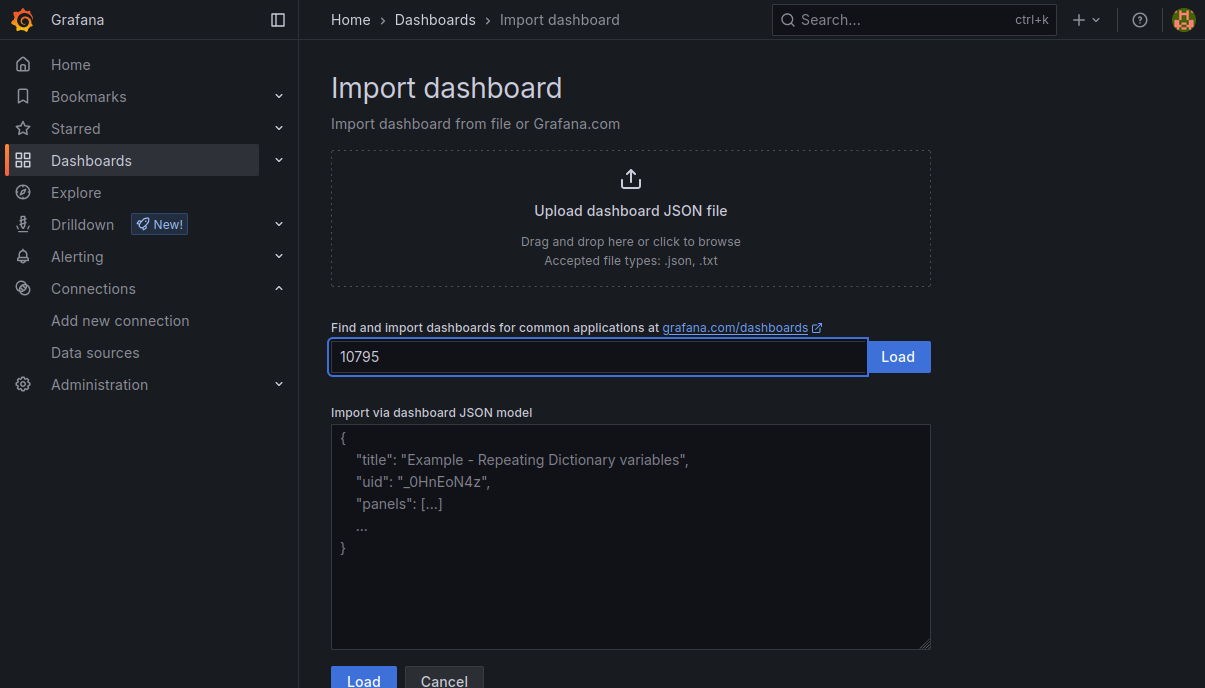
Настроим grafana-server для запуска при загрузке (рис. 23):

Рисунок 23 - Настройка grafana-server

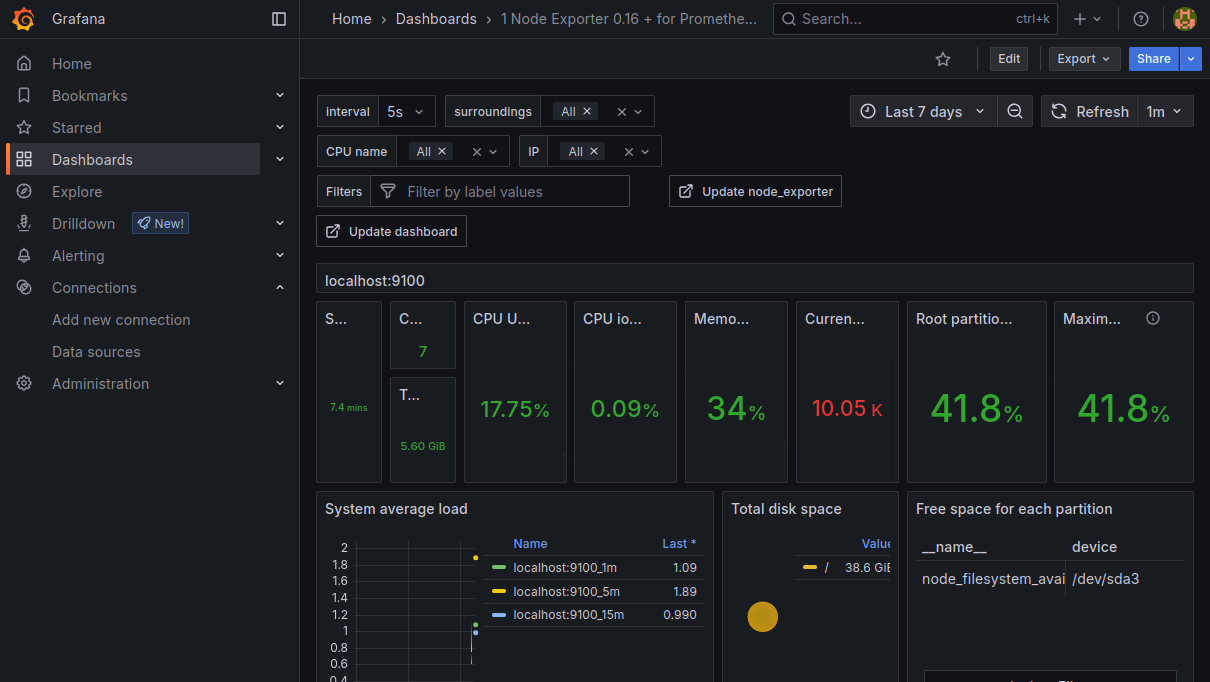
Настроим источники данных (рис. 24):

Рисунок 24 - Настройки источников данных

Используем готовый дашборд (рис. 25):

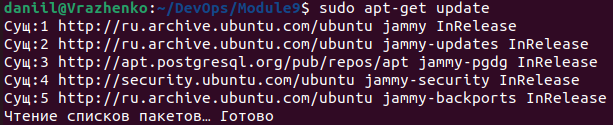
Рисунок 25 - Импорт готового дашборда

Откроем установленный дашборд (рис. 26):

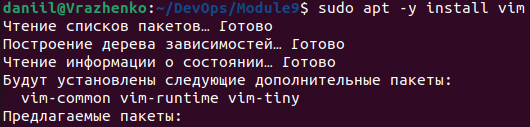
Рисунок 26 - Установленный дашборд

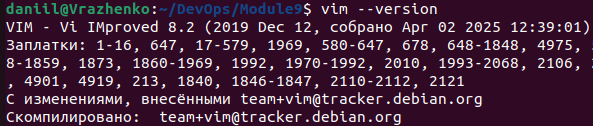
# Модуль 9. Виртуализация в DevOps

Обновим базу данных пакетов с помощью apt-get (рис. 27):

Рисунок 27 - Обновление базы данных пакетов

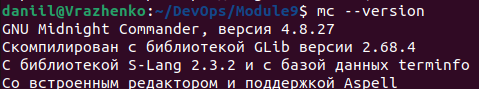
Установим vim и проверим версию (рис. 28-29):

Рисунок 28 - Установка vim

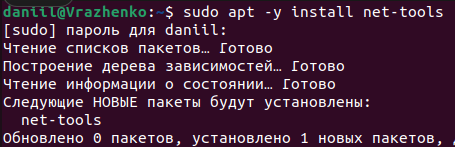
Рисунок 29 - Версия vim

Установим mc и проверим версию (рис. 30-31):

Рисунок 30 - Установка mc

Рисунок 31 - Версия mc

Установим net-tools (рис. 32):

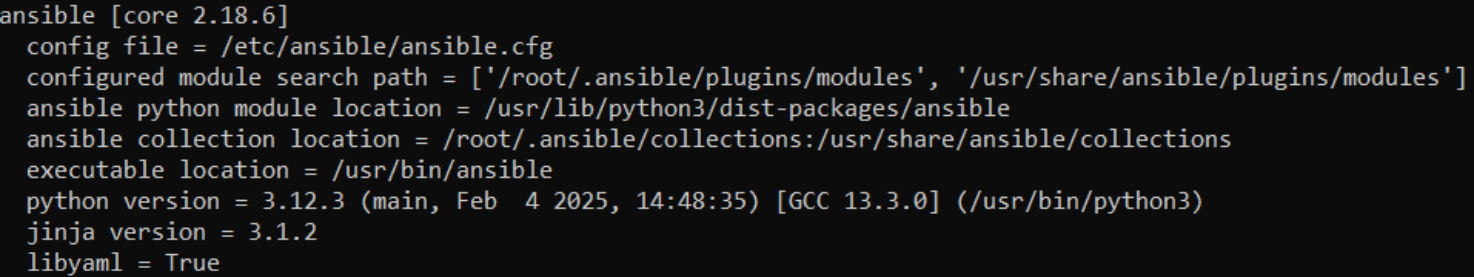
Рисунок 32 - Установка net-tools

К сожалению, проверить версию net-tools напрямую (как, у vim или mc) не получится, так как net-tools — это набор отдельных утилит, а не единая программа. Однако можно использовать флаг --version с некоторыми инструментами из этого пакета, например, ifconfig (рис. 33):

Рисунок 33 - Версия net-tools

# Модуль 12. Конфигурационное управление. Что такое IaC

Проверим установленную версию Ansible (рис. 34):

Рисунок 34 - Версия Ansible

Настроим конфигурационный файл (рис. 35):

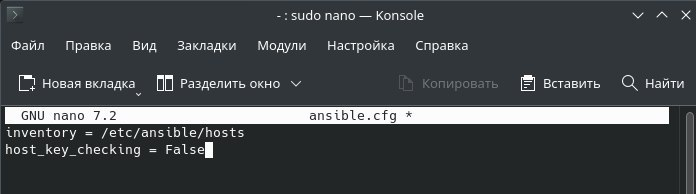


Рисунок 35 - Конфигурационный файл

Проверим доступность наших узлов (рис. 36):

Рисунок 36 - Доступность узлов

Создадим комплексный playbook (рис. 37-38):

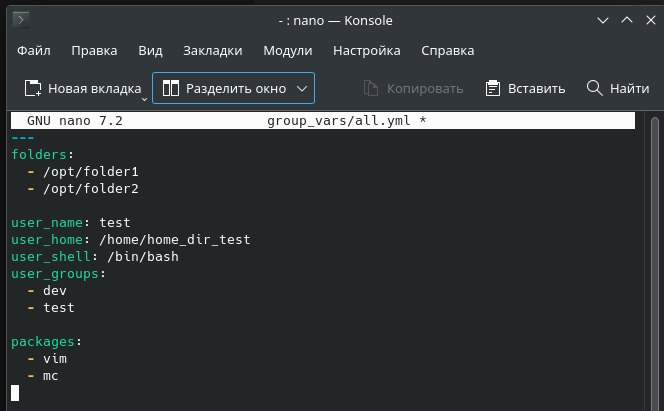
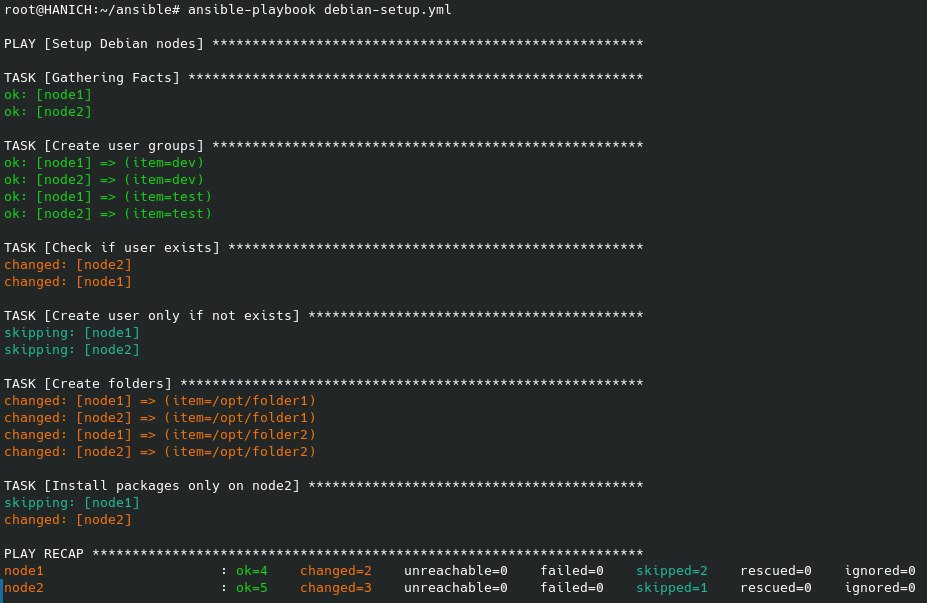
Рисунок 37 - Playbook (часть 1)

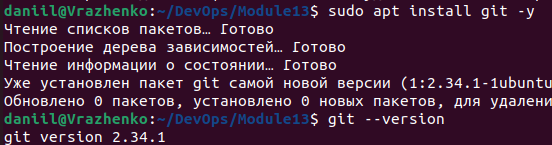
Рисунок 38 - Playbook (часть 2)

Применим playbook к целевым узлам (рис. 39):

Рисунок 39 - Результат запуска playbook

# Модуль 13. Системы контроля версий. Распределённая система управления версиями Git

Установим Git на сервер (рис. 40):

Рисунок 40 - Установка Git

Создадим пользователя git (рис. 41):

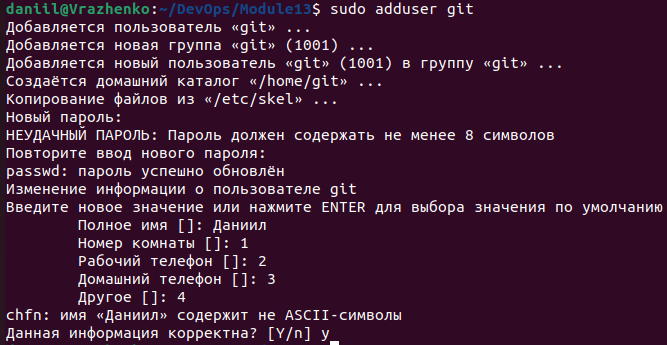


Рисунок 41 - Создание пользователя git

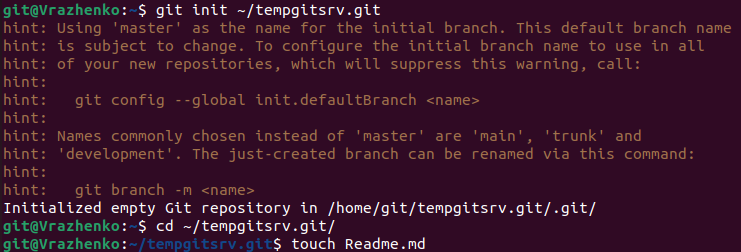
Логинимся под пользователя git (рис. 42):

Рисунок 42 - Логин под пользователя git

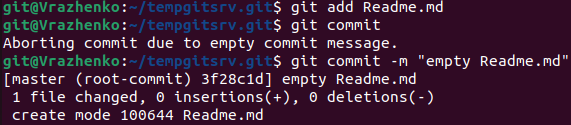
Предварительно настроим Git (рис. 43):

Рисунок 43 - Предварительные настройки Git

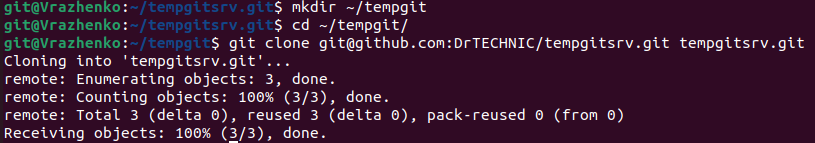
Инициализируем новый репозиторий и добавим в него пустой файл (рис. 44):

Рисунок 44 - Инициализация репозитория и добавление файла

Закоммитим изменение (рис. 45):

Рисунок 45 - Коммит изменения

Склонируем Git локально (рис. 46):

Рисунок 46 - Клонирование Git локально

Создадим новую ветвь (рис. 47):

Рисунок 47 - Создание новой ветви

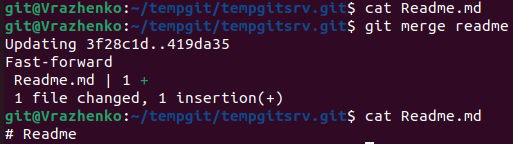
Создадим новый файл и отправим изменения в удаленный репозиторий (рис. 48):

Рисунок 48 - Отправление изменений в репозиторий

Выведем список всех веток на сервере (рис. 49):

Рисунок 49 - Список всех веток

Сольем ветки (рис. 50):

Рисунок 50 - Слияние веток

# Модуль 14. Жизненный цикл ПО

Установим Jenkins с помощью официальной инструкции (рис. 51-52):

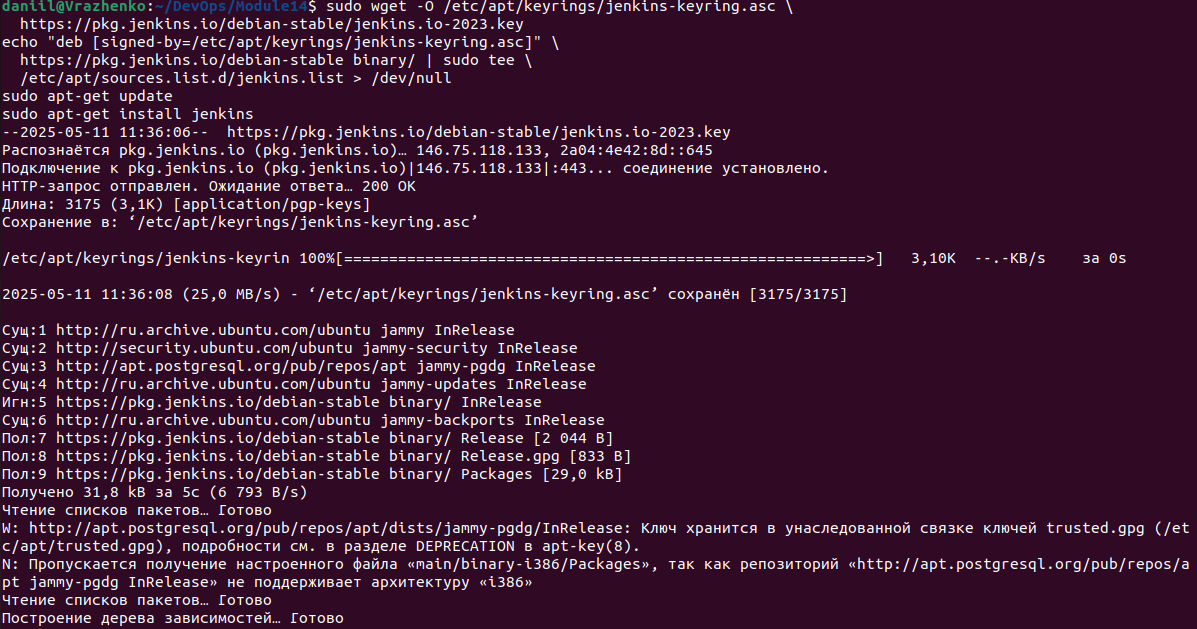


Рисунок 51 - Первая часть установки

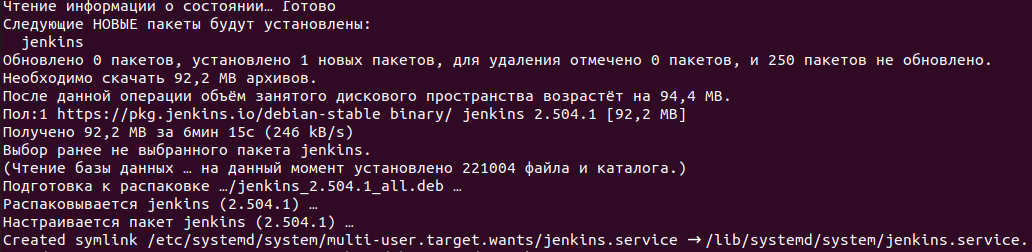
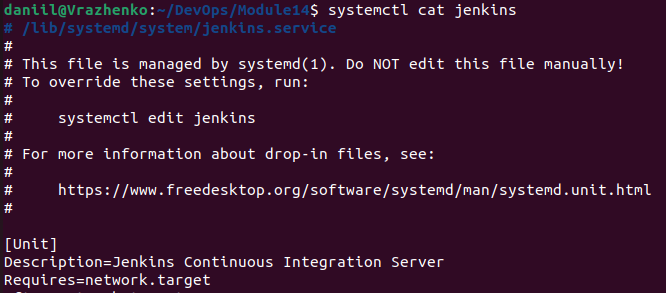


Рисунок 52 - Вторая часть установки

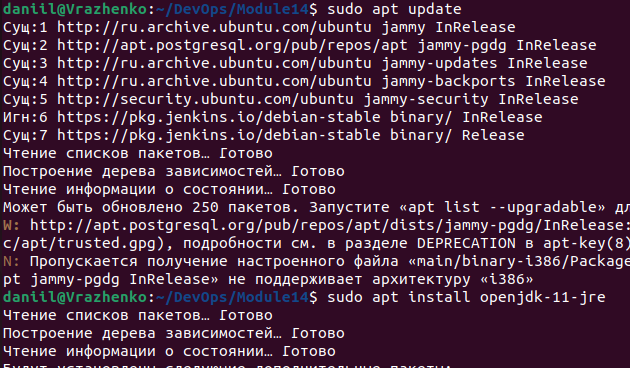
Настроим Jenkins как демон (рис. 53):

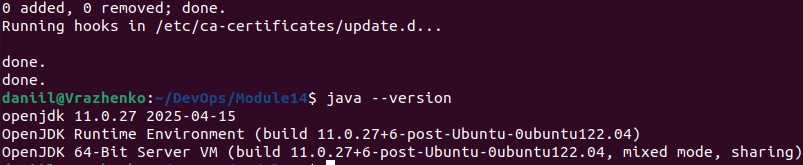
Рисунок 53 - Просмотр подробностей

Создадим пользователя для запуска службы (рис. 54):

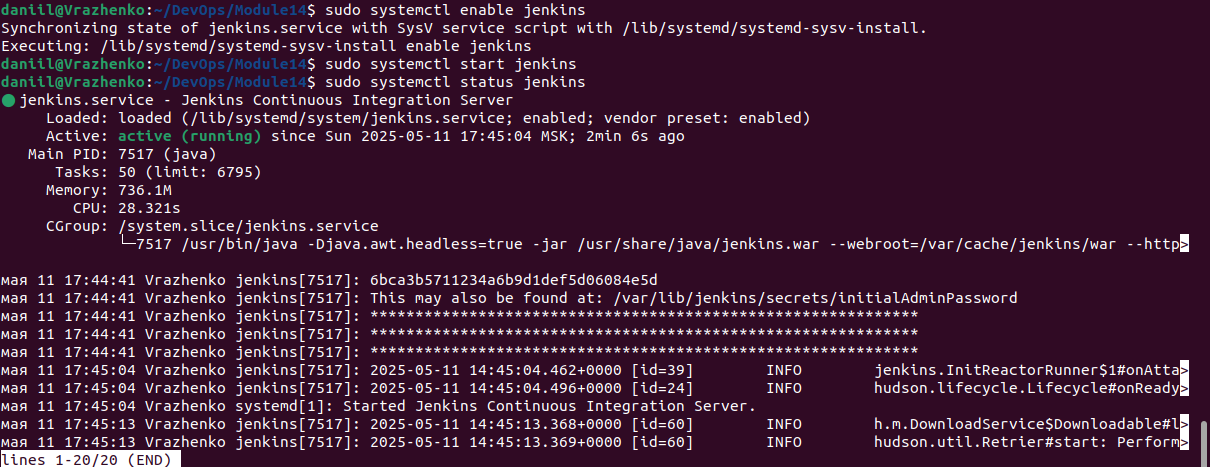
Рисунок 54 - id пользователя

Установим Java (рис. 55-56):

Рисунок 55 - Первая часть установки

Рисунок 56 - Вторая часть установки

Запустим Jenkins (рис. 57):

Рисунок 57 - Запуск Jenkins и его статус

Разблокируем Jenkins (рис. 58):

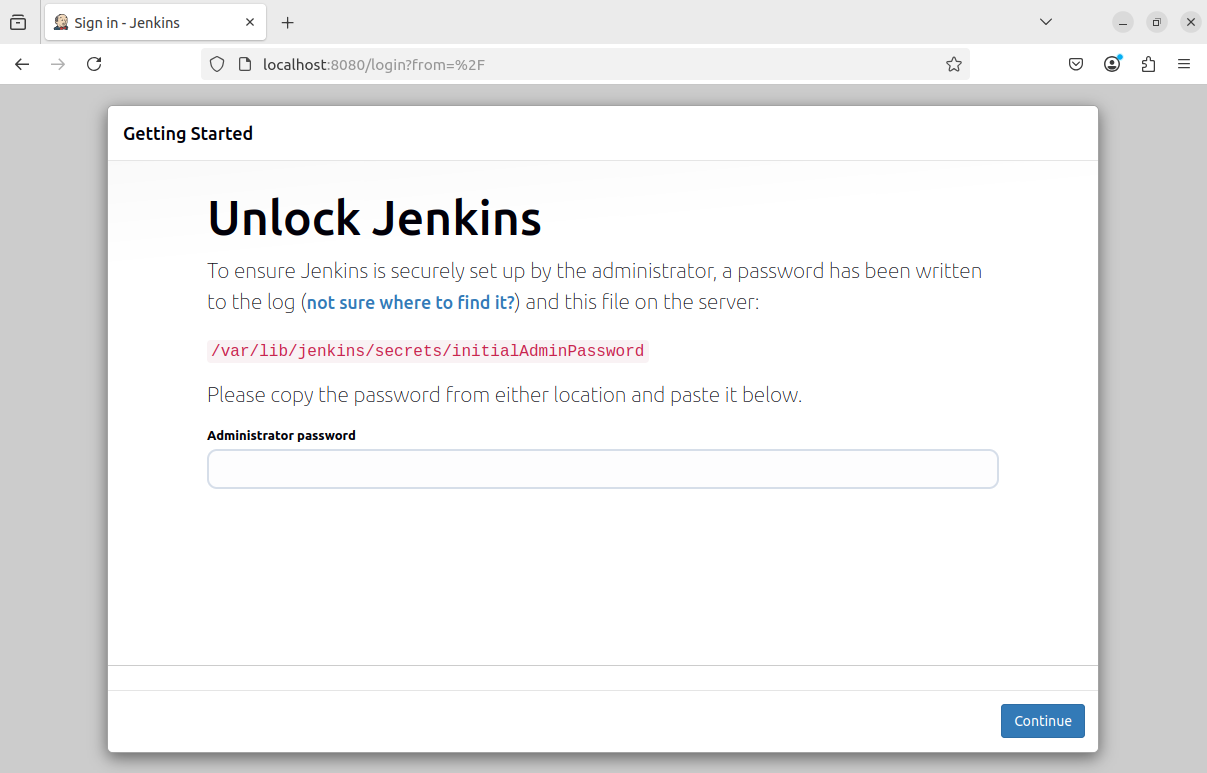


Рисунок 58 - Разблокировка Jenkins

Получим пароль для разблокировки Jenkins (рис. 59):

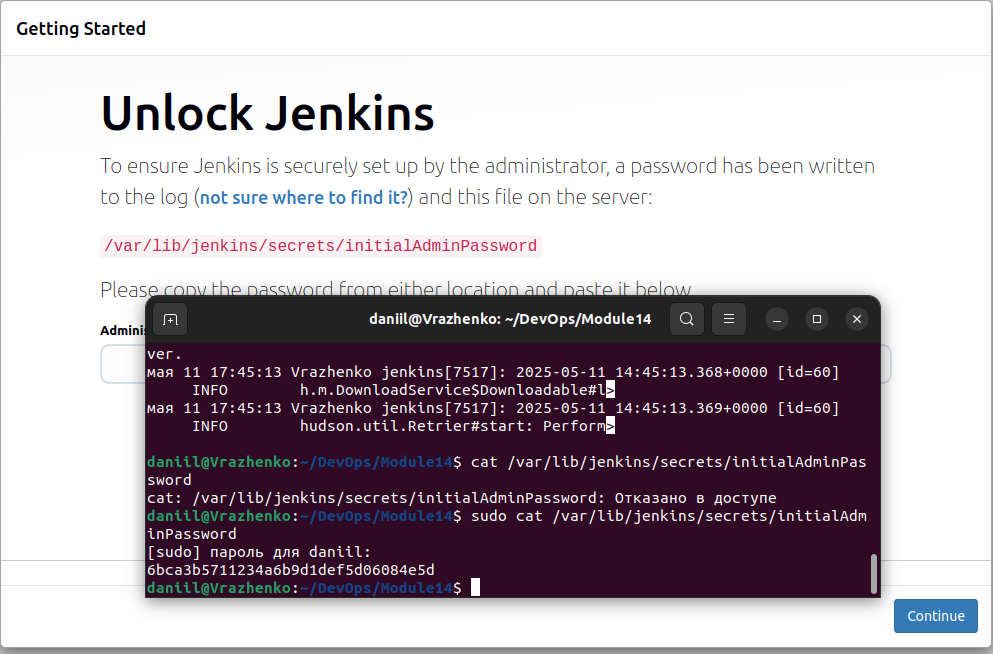
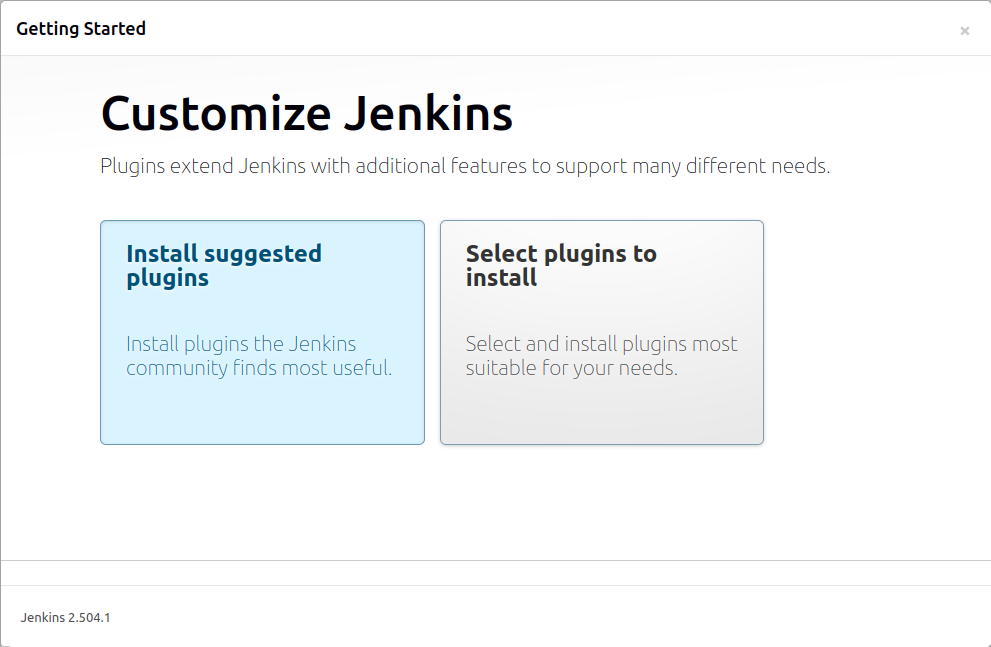
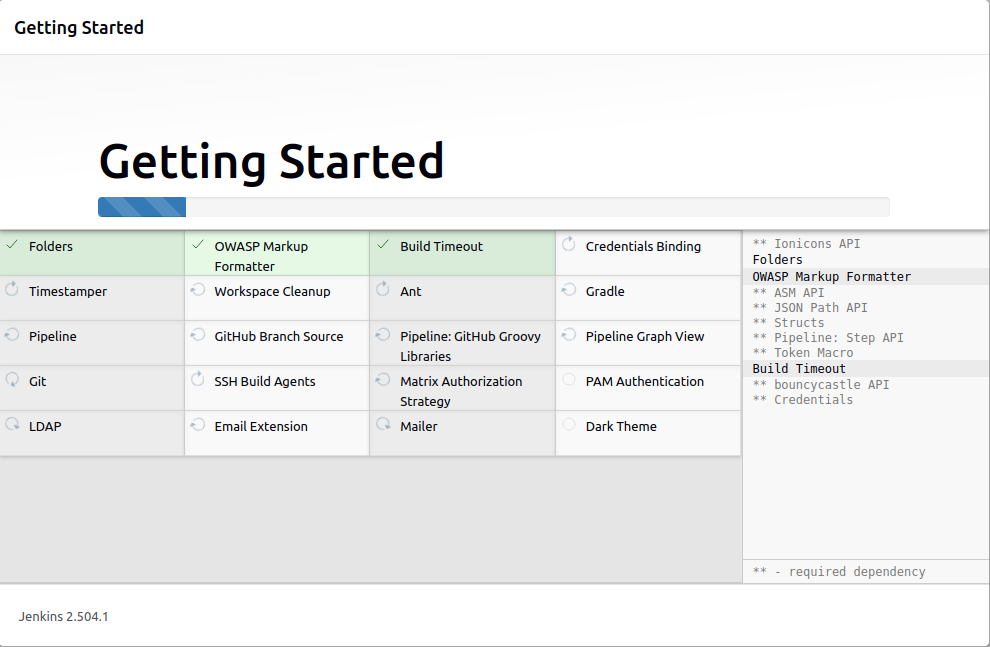


Рисунок 59 - Пароль для Jenkins

Установим плагины для Jenkins (рис. 60-61):

Рисунок 60 - Плагины для Jenkins

Рисунок 61 - Установка плагинов Jenkins

Создадим первого пользователя-администратора (рис. 62):

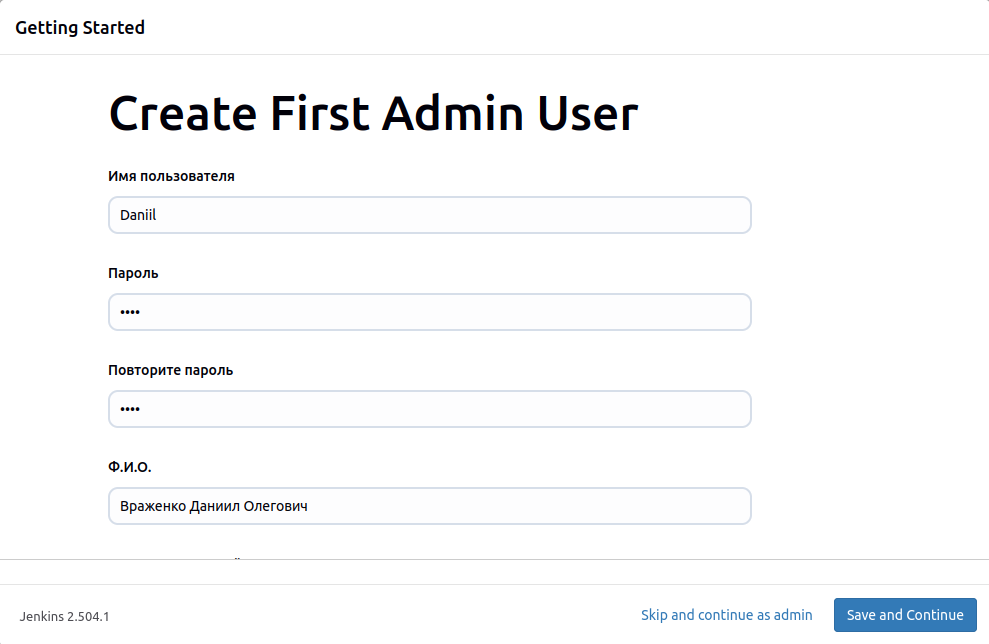
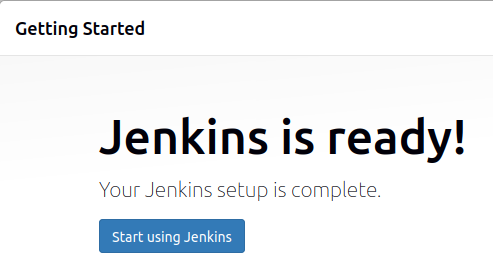
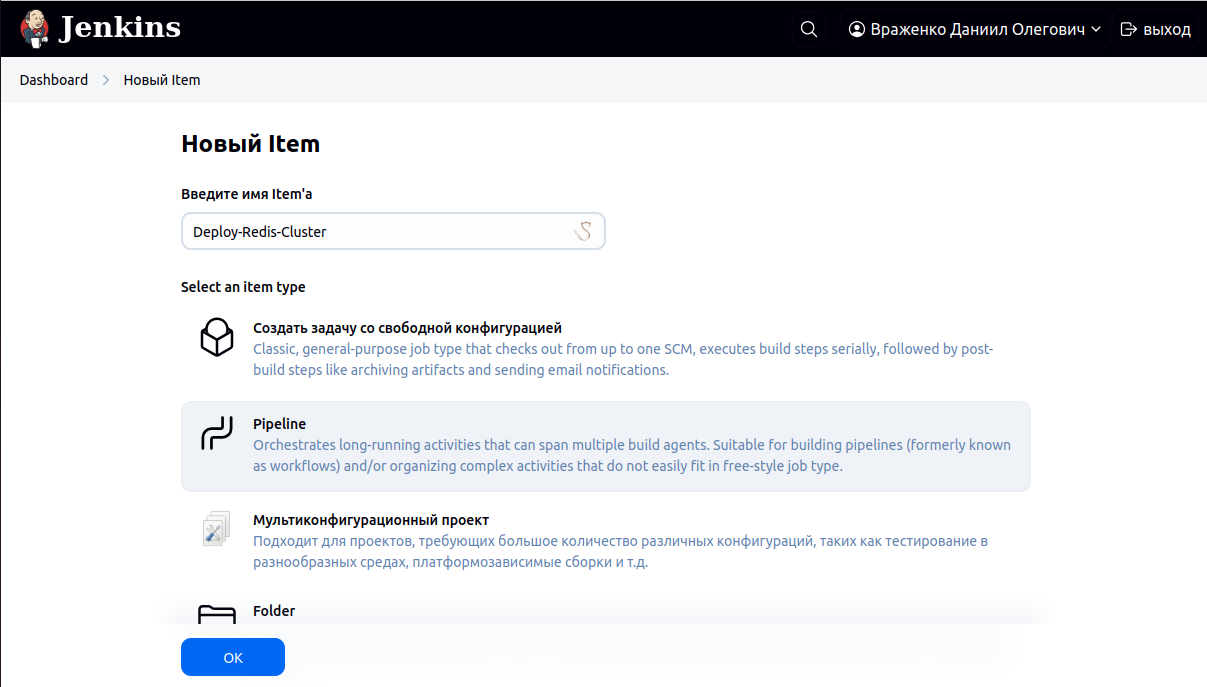


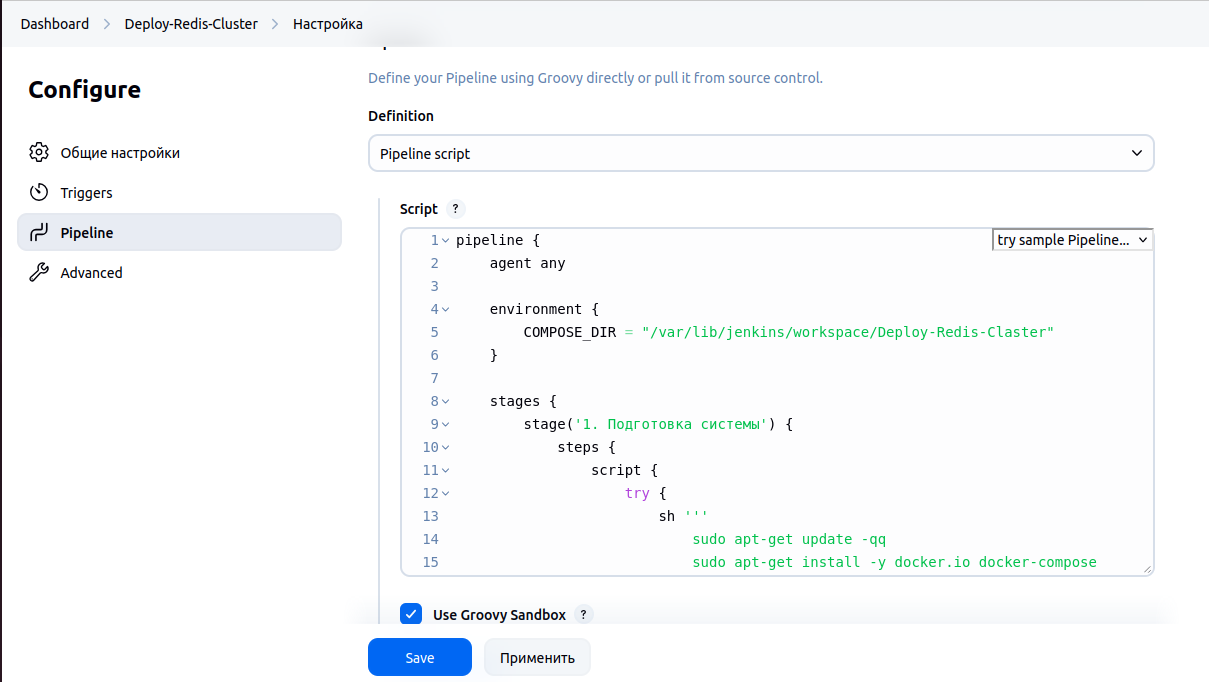
Рисунок 62 - Создание первого пользователя-администратора

Видим, что Jenkins готов к использованию (рис. 63):

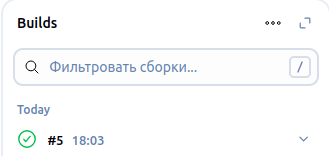
Рисунок 63 - Jenkins готов

Создадим Jenkins Pipeline и настроим его (рис. 64-65):

Рисунок 64 - Создание Pipeline

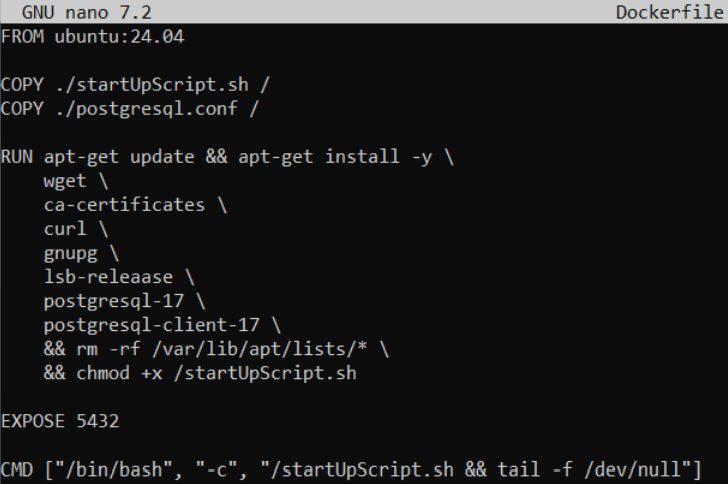
Рисунок 65 - Небольшая часть Pipeline скрипта

После нескольких попыток и изменений скрипта получаем успешный билд (рис. 66):

Рисунок 66 - Успешный билд

# Модуль 15. Практические навыки работы с Docker

Напишем dockerfile (рис.67):

Рисунок 67 - Содержимое dockerfile

Подготовим конфигурационный файл для корректной работы контейнера (рис. 68):

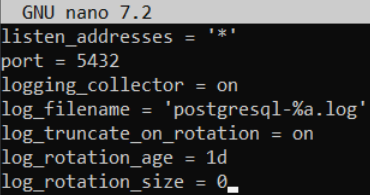
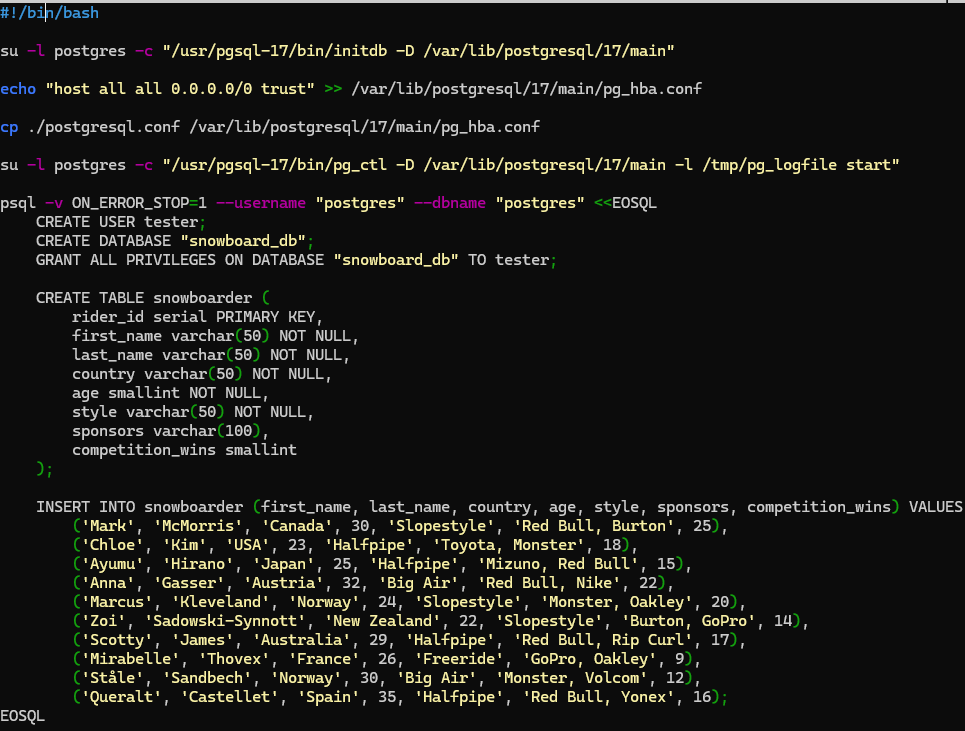


Рисунок 68 - Содержимое postgresql.conf

Создадим стартовый скрипт для автоматической инициализации базы данных при запуске (рис. 69):

Рисунок 69 - Содержимое startUpScript.sh

Выполним сборку докер-образа (рис. 70):

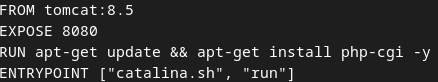
Рисунок 70 - Сборка образа

Запустим контейнер с пробросом портов и подключением volumes (рис. 71):

Рисунок 71 - Запуск контейнера

# Модуль 16. Микросервисы и микросервисная архитектура

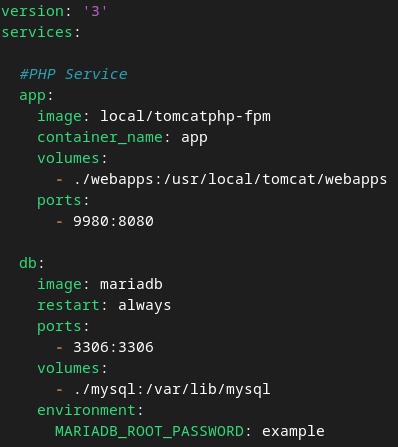
Напишем dockerfile (рис. 72):

Рисунок 72 - Содержимое Dockerfile

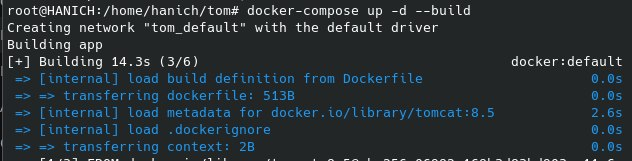
Создадим образ (рис. 73):

Рисунок 73 - Успешная сборка образа

Напишем docker compose файл (рис. 74):

Рисунок 74 - Содержимое docker-compose.yml

Запустим сервис (рис. 75):

Рисунок 75 - Результат docker-compose

Напишем test.php (рис. 76):

Рисунок 76 - Содержимое test.php

Перейдем на сайт (рис. 77):

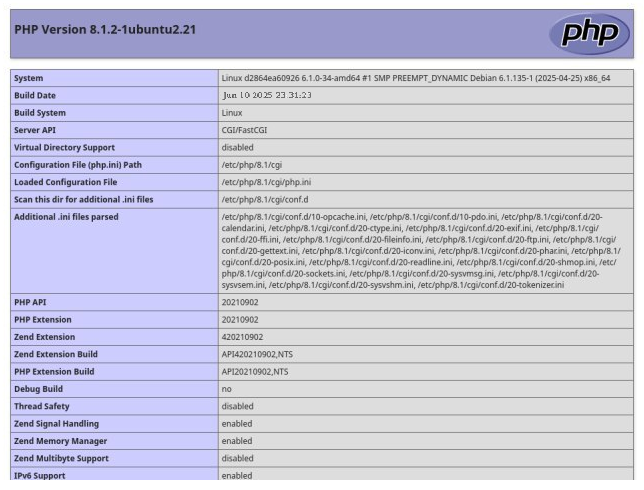
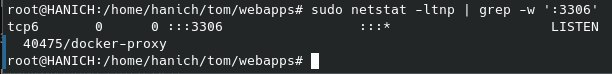


Рисунок 77 - Сайт

Просмотрим доступность порта MariaDB (рис. 78):

Рисунок 78 - Порт MariaDB

# ****Заключение****

Мы проделали работу по следующим модулям:

**Модуль 6. Администрирование баз данных:**

* Установили PostgreSQL 12 на Ubuntu.
* Проверили работоспособность базы данных.
* Подключились к PostgreSQL через командную строку.
* Создали пользователя и базу данных.
* Сформировали таблицу, заполнили её данными и проверили корректность внесённых записей.
* Дополнили таблицу новыми данными и выполнили повторную проверку.

**Модуль 8. Мониторинг, логирование и оповещение событий:**

* Загрузили и распаковали Node Exporter, сделали его исполняемым и запустили на порту 9100.
* Убедились, что метрики успешно экспортируются.
* Установили Prometheus, настроили его конфигурационный файл prometheus.yml и запустили сервер.
* Развернули Grafana с помощью пакета .deb, настроили автозапуск сервера.
* Добавили Prometheus в качестве источника данных в Grafana.
* Импортировали готовый дашборд и проверили отображение метрик.

**Модуль 9. Виртуализация в DevOps:**

* Обновили базу пакетов Ubuntu.
* Установили и проверили версии утилит:
  + Vim
  + Midnight Commander (mc)
  + Net-tools

**Модуль 12. Конфигурационное управление. Что такое IaC:**

* Проверили установленную версию Ansible.
* Настроили конфигурационный файл для работы с целевыми узлами.
* Протестировали доступность управляемых серверов.
* Разработали комплексный playbook для автоматизации конфигурации.
* Применили playbook к целевым узлам для выполнения задач управления.

**Модуль 13. Системы контроля версий. Git:**

* Установили Git на сервер.
* Создали и переключились на пользователя git.
* Настроили глобальные параметры Git (имя пользователя и email).
* Инициализировали новый репозиторий и добавили в него файл.
* Создали первый коммит.
* Склонировали репозиторий на локальную машину.
* Создали новую ветку, добавили файл и отправили изменения в удалённый репозиторий.
* Просмотрели список всех веток на сервере.
* Выполнили слияние веток.

**Модуль 14. Жизненный цикл ПО:**

* Установили Jenkins, следуя официальной инструкции.
* Настроили Jenkins как демона, создали системного пользователя для его работы.
* Установили Java.
* Запустили Jenkins.
* Установили рекомендуемые плагины.
* Создали административного пользователя и завершили первоначальную настройку.

**Модуль 15. Практические навыки работы с Docker:**

* Разработали Dockerfile для создания кастомного образа.
* Подготовили конфигурационные файлы для корректной работы контейнера.
* Создали стартовый скрипт для автоматической инициализации БД.
* Выполнили сборку Docker-образа на основе Dockerfile.
* Запустили контейнер с пробросом портов и подключением томов (volumes).

**Модуль 16. Микросервисы и микросервисная архитектура:**

* Разработали Dockerfile для микросервисного приложения.
* Создали Docker-образ на основе конфигурации.
* Написали docker-compose.yaml для оркестрации многоконтейнерного приложения.
* Запустили сервисы через Docker Compose.
* Создали тестовый PHP-скрипт для проверки работы.
* Проверили доступность веб-интерфейса и соединение с СУБД MariaDB.

# ****Список литературы****

1. Сельский, М. В. *PostgreSQL. Основы администрирования.* — СПб.: Питер, 2021. — 320 с.
2. Хайт, Дж. *DevOps: практика и теория.* — М.: Эксмо, 2022. — 288 с.
3. Неофитов, А. В. Apache Kafka: потоковая обработка данных. — М.: ДМК Пресс, 2023. — 256 с.
4. Лукьянова, Е. С. *Kubernetes: оркестрация контейнеров.* — СПб.: БХВ, 2022. — 384 с.
5. Петров, И. К. *Микросервисная архитектура: принципы и паттерны.* — М.: Инфра-Инженерия, 2020. — 412 с.
6. Смирнов, П. А. *Анализ больших данных в сельском хозяйстве.* — М.: Агропромиздат, 2023. — 198 с.
7. Документация PostgreSQL. — <https://www.postgresql.org/docs/>
8. Васильев, Д. Р. *Цифровая трансформация госсектора.* — М.: Альпина Паблишер, 2021. — 304 с.
9. Громов, А. Ю. *Интеграция ведомственных информационных систем.* — М.: ИТ-Академия, 2022. — 176 с.
10. Козлов, В. И. *Tarantool: высоконагруженные базы данных.* — СПб.: Питер, 2023. — 224 с.