Лабораторная работа №5.

Управление ресурсами и изоляция с помощью cgroups, namespaces. Контейнеры

Выполнила студентка группы М3311

Авсюкевич Анастасия

**Задачи:**

1. Научиться создавать, изменять и удалять лимиты и квоты для пользователей и процессов с помощью Control groups (cgroups

V1/V2).

2. Научиться настраивать изоляцию различных ресурсов ОС с помощью namespaces.

3. Научиться работать с контейнерами и понимать, как данная технология работает «под капотом»

**Задания:**

**Мой ID - 84**

1. Квоты на процессор для конкретного пользователя (cgroups v2)

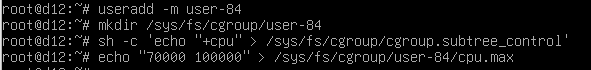
• Создайте пользователя: user-ID (например, user-72).

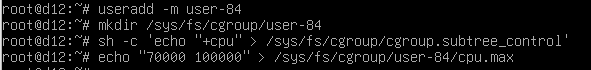
• Назначьте квоту процессора на основе номера пользователя:

◦ Если имя пользователя заканчивается на 0-30: 30% CPU.

◦ Если имя пользователя заканчивается на 31-70: 50% CPU.

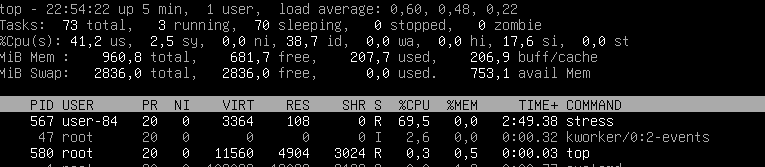
◦ Если имя пользователя заканчивается на 71-99: 70% CPU.











2. Ограничение памяти для процесса (cgroups)

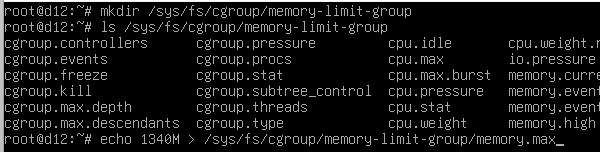
• Создайте cgroup для ограничения памяти, потребляемой процессом.

• Запустите процесс и переместите его в созданную вами группу.

◦ Пример команды: tail /dev/zero.

• Ограничьте потребление памяти следующим образом: ID\*10 + 500 МБ (например, ID=23 → 730 МБ).

• Проверьте, что при исчерпании памяти процессом он прерывается ОС.







3. Ограничение дискового ввода-вывода для сценария резервного копирования (cgroups)

• Скрипт резервного копирования (backup.sh) перегружает дисковую подсистему.

• Ограничьте его до:

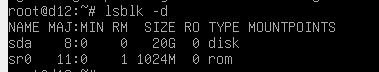
◦ Чтение: 1000 + <ID>\*10 IOPS.

◦ Запись: 500 + <ID>\*10 IOPS.

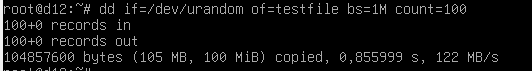
• Используйте cgcreate для установки ограничений io.max.

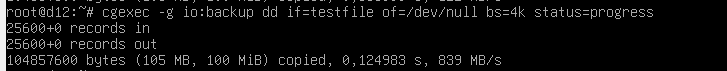
• Протестируйте с помощью fio или dd.













Скорость чтения и записи не превышена.

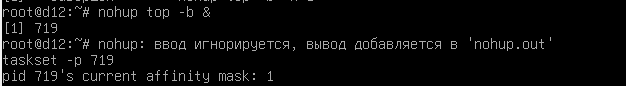
4. Закрепление к определенному ядру процессора для приложения

• Настройте с помощью cgroups процесс команды top за процессором 0.

• Используйте cpuset.cpus в cgroups.

• Проверьте с помощью taskset -p <PID>. (требуется пакет sysstat)





Здесь 1 в двоичной системе (0001) означает, что процесс может работать только на первом процессоре (процессоре 0), значит все ок.

5. Динамическая корректировка ресурсов (cgroups)

• Напишите сценарий для мониторинга нагрузки по CPU и динамического изменения cpu.max определенного процесса (его

идентификатор задается как входной параметр скрипта).

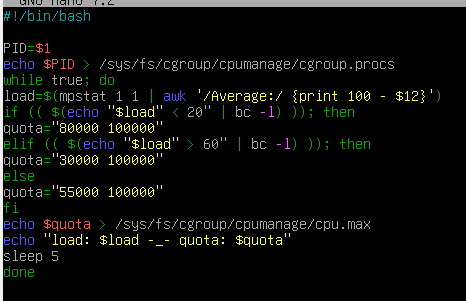
• Квота ЦП для процесса должна регулироваться в зависимости от общей нагрузки на систему:

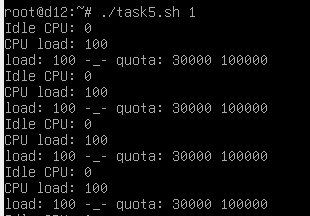
◦ Низкая нагрузка (CPU < 20%): 80% CPU.

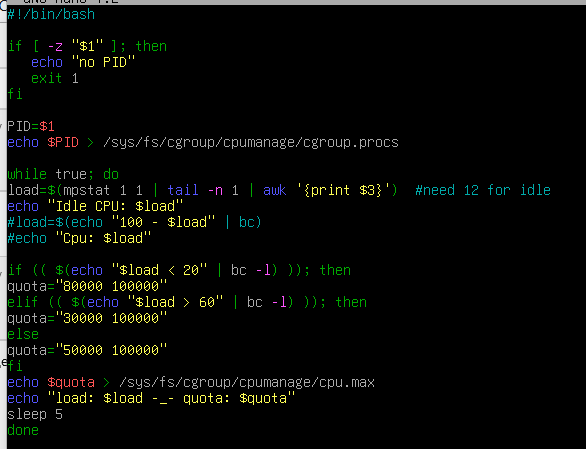
◦ Высокая нагрузка (CPU > 60%): 30% CPU.

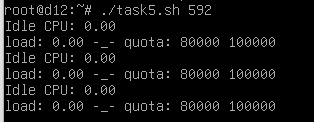


Task5.sh:



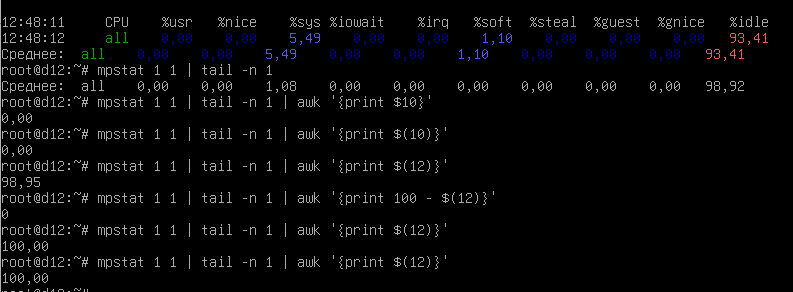




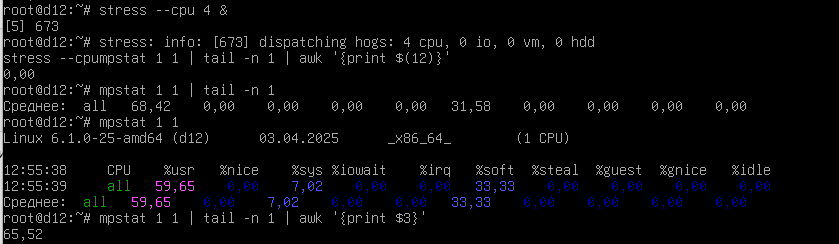


Пруф:

Почти все время простаивает процессор, поэтому и idle почти 100

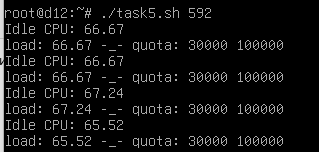


Создаем имитацию нагрузки, тут отображается нагрузка процессора но для пользовательских процессов



Pid: 673

Тут можно увидеть, что скрипт отрабатывает правильно, но считается сейчас по %usr



6. Создание изолированного имени хоста (пространство имен UTS)

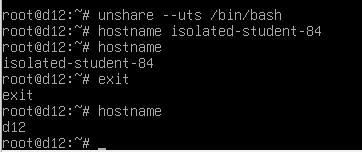
• Запустите оболочку (shell/bash) в отдельном namespace, в которой можно изменить имя хоста, не затрагивая хост.

• Измените имя хоста внутри пространства имен на isolated-student-<ID>.

• Проверьте изоляцию:

◦ hostname # Должно отображаться «isolated-student-».

◦ Проверьте имя хоста (в новом терминале): hostname # По-прежнему показывает оригинальное имя хоста.



--uts — это флаг, который указывает, что процесс будет изолирован в новом **UTS (UNIX Time-sharing System) namespace**. Это пространство имен отвечает за изоляцию имени хоста и домена. Это позволяет процессам в одном пространстве имен иметь уникальные имена хостов, которые не будут изменять или воздействовать на имя хоста в основном (глобальном) пространстве имен.

7. Изоляция процессов (пространство имен PID)

• Создайте пространство имен, в котором процессы хоста будут невидимы:

◦ unshare --pid --fork bash.

• Смонтируйте новый каталог /proc:

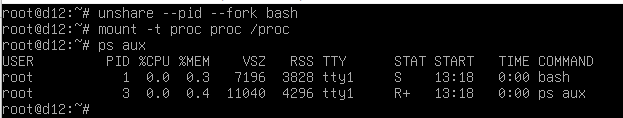
◦ mount -t proc proc /proc.

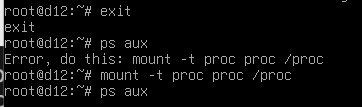
• Проверьте процессы:

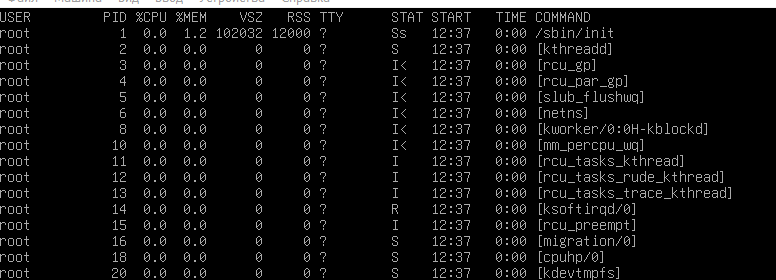
◦ ps aux # Показывает только 2-3 процесса (например, bash, ps).

• Сравните с хостом (в новом терминале):

◦ ps aux # Показывает все процессы хоста.







8. Изолированная файловая система (пространство имен Mount)

• Создайте каталог, видимый только в пространстве имен:

◦ unshare --mount bash.

• Создайте приватный каталог:

◦ mkdir /tmp/private\_$(whoami).

• Смонтируйте временную файловую систему:

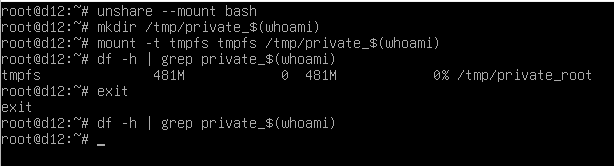
◦ mount -t tmpfs tmpfs /tmp/private\_$(whoami).

• Проверьте изоляцию:

◦ df -h | grep private\_$(whoami) # Запишите в отчет результат.

• Проверка на хосте (в новом терминале):

◦ df -h | grep private\_$(whoami).



9. Отключение доступа к сети (пространство имен Network)

• Запустите командный интерпретатор bash без доступа к сети.

• Проверьте сетевые интерфейсы:

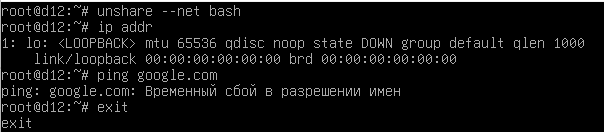
◦ ip addr # Запишите в отчет, что показывает команда.

• Проверьте подключение:

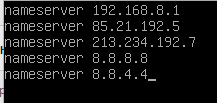
◦ ping google.com.

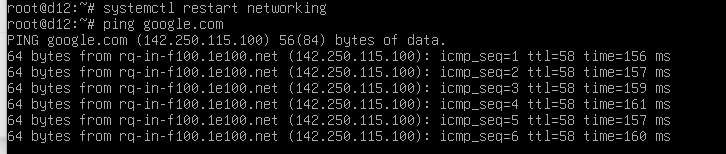
• Сравните с хостом (в новом терминале):

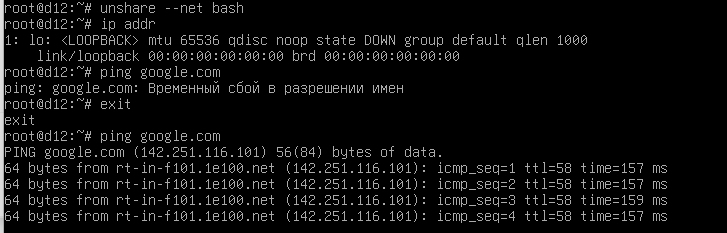
◦ ping google.com.



Nano /etc/resolv.conf





После повторной все было ок:  


10. Создайте и проанализируйте монтирование OverlayFS

Шаги:

a. Первоначальная настройка:

• Создайте каталоги:

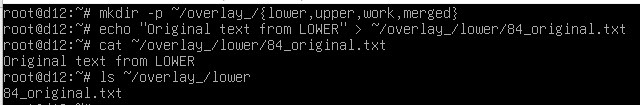
mkdir -p ~/overlay\_/{lower,upper,work,merged}

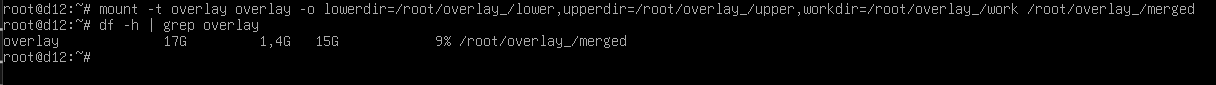
• В каталоге lower создайте файл с именем <ID>\_original.txt с содержанием:

Оригинальный текст из LOWER

• Смонтируйте OverlayFS:

mount -t overlay overlay -o lowerdir=~/overlay\_/lower,upperdir=~/overlay\_/upper,workdir=~/overlay\_/work ~/overlay\_/merged





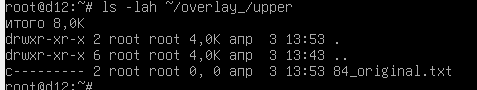
b. Имитация неполадки и отладка:

• Удалите файл <ID>\_original.txt из каталога merged.

• Понаблюдайте: Какой файл(ы) появился(ись) в верхнем каталоге? Задокументируйте их имена и содержимое.

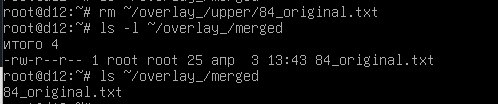
• Измените каталог merged, чтобы восстановить <ID>\_original.txt, не размонтируя и не изменяя нижний уровень.





Восстановление в merged:

Чтобы восстановить исходный файл, нужно **удалить whiteout-метку** из upper, тогда файл снова станет видимым в merged (так как OverlayFS снова начнет показывать его из lower).



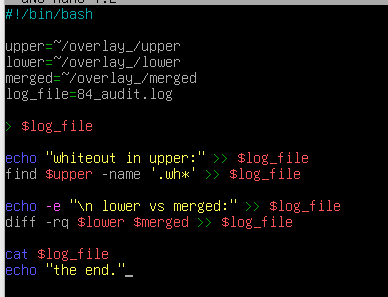
При удалении файла в каталоге merged, **OverlayFS** не удаляет его физически из нижнего слоя (lower). Вместо этого в верхнем каталоге (в каталоге upper) будет создан **whiteout файл**, который скрывает файл в объединённом слое.

c. Разработайте скрипт, который:

• Обнаруживает все whiteout файлы в верхнем каталоге upper.

• Сравнивает содержимое нижнего и объединенного для выявления несоответствий.

• Выводит отчет с именем <ID>\_audit.log.

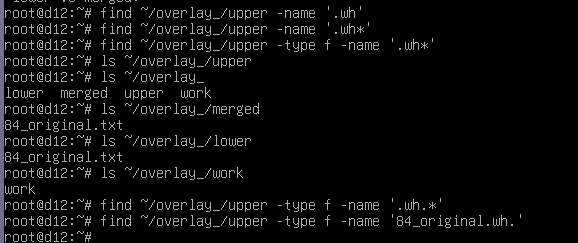


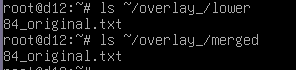
Или:

# Поиск whiteout-файлов (обычно это файлы с размером 0 или char устройство 0:0)

find ~/overlay\_/upper -type f -size 0 -o -type c -exec ls -l {} + > 84\_audit.log

find ~/overlay\_/upper -type c -exec stat -c "%n (device %t:%T)" {} + >> <84\_audit.lo





d. Ответьте на вопросы:

• Как OverlayFS скрывает файлы из нижнего слоя при удалении в объединенном?

Когда OverlayFS объединяет два каталога (например, нижний и верхний слои), он создает виртуальную файловую систему, которая позволяет работать с файлами, находящимися в этих слоях, как с единым каталогом.Когда файл удаляется в объединенной файловой системе, OverlayFS не удаляет его непосредственно из нижнего слоя. Вместо этого он использует механизм копирования на запись (copy-on-write, COW). Вот как это работает:

* Если файл присутствует в верхнем слое, а вы его удаляете, OverlayFS не удаляет файл физически. Вместо этого он помечает его как удаленный в верхнем слое, но файл остается в нижнем слое, скрытым от пользователя.
* Если файл отсутствует в верхнем слое, OverlayFS отображает его из нижнего слоя. Таким образом, файл не удаляется из нижнего слоя, а просто скрывается.

Суть в том, что физически файл продолжает существовать в нижнем слое, но благодаря механизмам OverlayFS, он скрывается, когда его удаляют в верхнем слое.

• Если вы удалите рабочий каталог work, сможете ли вы перемонтировать оверлей? Объясните, почему.

Каталог **work** в OverlayFS используется для хранения промежуточных данных, которые необходимы для функционирования файловой системы (например, временные файлы для копирования на запись).

Если вы удалить его, нарушится работа OverlayFS, так как этот каталог является необходимым для нормальной работы системы объединения слоев. Если каталог work будет удален или станет недоступен, OverlayFS не сможет функционировать должным образом, и попытка перемонтировать файловую систему приведет к ошибке.

* Каталог work используется для промежуточных операций при работе с файлами в верхнем слое.
* Без этого каталога система не может отслеживать изменения и преобразования, такие как копирование файлов на запись (COW). Это вызовет ошибку при попытке перемонтировать файловую систему или выполнить другие операции.

• Что произойдет с объединенным слоем, если верхний каталог будет пуст?

Если верхний каталог (upper layer) в OverlayFS пустой, то файловая система будет вести себя следующим образом:

* Файлы, которые присутствуют только в нижнем слое (lower layer), будут видимы в объединенной файловой системе.
* Файл не будет копироваться в верхний слой, так как верхний слой пуст. Все изменения или добавления файлов будут записываться только в верхний слой.
* Если в верхнем слое нет изменений, то фактически для пользователя будет видно только содержимое нижнего слоя.

Что происходит при этом:

* Все файлы из нижнего слоя отображаются в объединенной файловой системе, так как OverlayFS по умолчанию отображает содержимое нижнего слоя в объединенной файловой системе, если верхний слой пуст.
* Все операции записи (например, создание нового файла или изменение существующего) будут происходить в верхнем слое, а нижний слой останется неизменным.

11. Оптимизируйте Dockerfile для приведенного ниже приложения app.py

from flask import Flask

import socket

import os

app = Flask(\_\_name\_\_)

@app.route('/')

def container\_info():

# Get container IP

hostname = socket.gethostname()

ip\_address = socket.gethostbyname(hostname)

# Get student name from environment variable

student\_name = os.getenv('STUDENT\_NAME', 'Rincewind')

return f"Container IP: {ip\_address} Student: {student\_name}"

if \_\_name\_\_ == '\_\_main\_\_':

app.run(host='0.0.0.0', port=5000)

Исходный Dockerfile:

FROM python:latest

COPY . /app

WORKDIR /app

RUN pip install flask

CMD ["python", "app.py"]

Улучшите Dockerfile с учетом лучших практик:

• Используйте меньший базовый образ.

• Зафиксируйте версию образа.

• Запуск от имени пользователя, не являющегося root.

• Используйте кэширование слоев для зависимостей.

• Добавьте файл .dockerignore.

Сначала весь этот ужас для установки:

# Удаляем старые версии (если были)

sudo apt remove docker docker-engine docker.io containerd runc

sudo apt update

sudo apt install -y apt-transport-https ca-certificates curl gnupg lsb-release

# Добавляем ключ Docker

curl -fsSL https://download.docker.com/linux/debian/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

# Добавляем репозиторий Docker

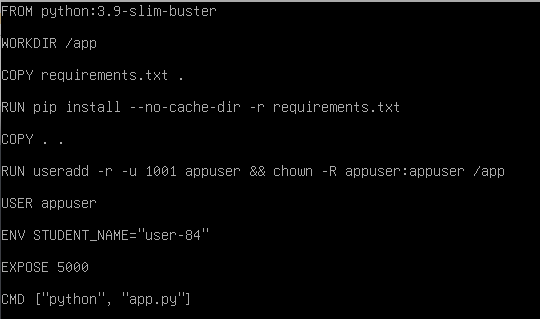
echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/debian $(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

# Устанавливаем Docker

sudo apt update

sudo apt install -y docker-ce docker-ce-cli containerd.io

Dockerfile:



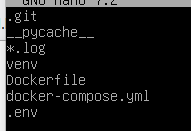
Комментарии к улучшениям:

1. Образ: Использован python:3.9-slim-buster (меньше базового образа).
2. Безопасность: Запуск от имени непривилегированного пользователя (appuser).
3. Кэширование: Сначала копируется requirements.txt, потом остальные файлы.
4. Чистота: Добавлен .dockerignore, чтобы не копировать мусор.
5. Детерминизм: Фиксированные версии Python и Flask.

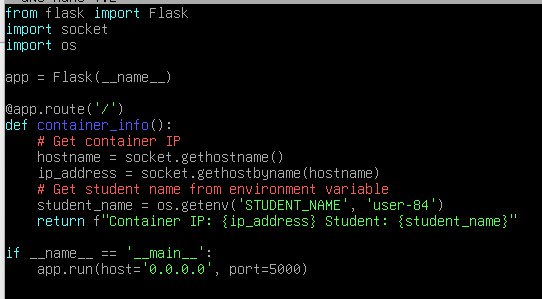
Requirements.txt:



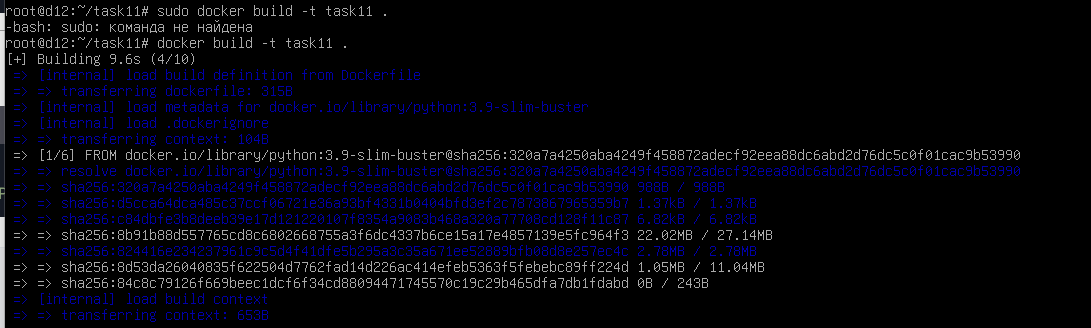
.dockerignore:



App.py:

****

Запуск:





* -d — запуск в фоне (демон)
* --rm — автоматически удалить контейнер после остановки

12. Установка платформы публикации WordPress с помощью Docker Compose

Задача:

Создать docker-compose.yml для запуска WordPress и MySQL/MariaDB с сохранением состояния при перезапуске контейнеров.

Используйте:

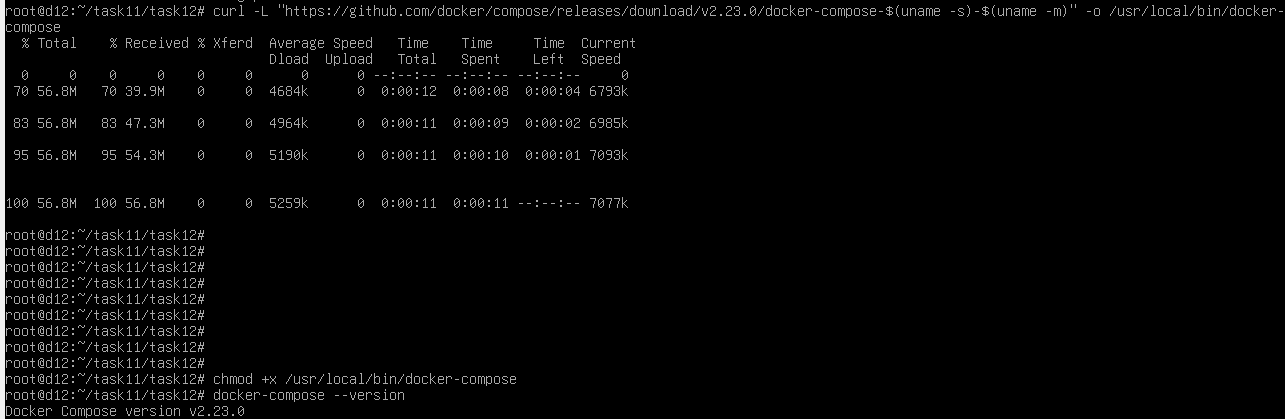
• Порт <ID>+2000 для WordPress (например, ID = 65 → port = 2065).

• Пароль базы данных: [ваше\_имя]\_db\_pass.

• Том с именем [ваше\_имя]-wp-data для WordPress.

Установка:







Docker-compose.yml:

