Лабораторная работа №5

Управление ресурсами и изоляция с помощью cgroups, namespaces. Контейнеры

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».

Подсказки:

1. В работе используются следующие команды (в дополнение к изученным в предыдущих лабораторных работах):
   - `cgcreate`, `cgset`, `cgexec`, `systemctl set-property`, `systemd-cgtop`, `unshare`, `nsenter`, `pivot_root`, `chroot`, `docker run`, `docker build`, `docker exec`, `docker inspect`, `docker volume`, `docker network`, `docker compose`, `stress`, `stress-ng`, `mpstat`, `debootstrap`.
2. Прочитайте документацию по указанным командам и определите их назначение до начала выполнения заданий.
3. Для получения подробного справочного руководства по любой команде можно набрать в консоли:
   - `man <название команды>`
   - Для краткой справки: `<название_команды> -h` или `<название_команды> --help`.
4. Для выполнения части заданий понадобится установить:
   - `sysstat`, `stress`, `docker`.

Задание:

Создайте текстовый файл, в котором запишете последовательность команд для выполнения каждого из нижеследующих заданий. Для команд, имеющих интерактивный интерфейс, опишите последовательность выбора управляющих команд и их параметров. Если решение заключается в изменении конфигурационного файла, укажите название файла и вносимые правки. Во многих заданиях будет фигурировать **ID** – это последние 2 цифры вашего номера студента в ИСУ.

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`.

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`)

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.

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.

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`.

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.

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`.

d. Ответьте на вопросы:

- Как OverlayFS скрывает файлы из нижнего слоя при удалении в объединенном?
- Если вы удалите рабочий каталог `work`, сможете ли вы перемонтировать оверлей? Объясните, почему.
- Что произойдет с объединенным слоем, если верхний каталог будет пуст?

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`.

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.