# Middle DevOps - Linux

## 1. Опишіть архітектуру ядра Linux.

Ядро Linux має модульну архітектуру, яка дозволяє додавати та видаляти функціональність у вигляді модулів. Основні компоненти архітектури ядра Linux включають:

*Системні виклики (System Calls):*

Ядро забезпечує механізми для взаємодії зі зовнішніми програмами через системні виклики. Це дозволяє програмам звертатися до ресурсів системи, таких як файлова система, мережеві можливості та інші.

*Менеджер пам'яті (Memory Management):*

Ядро відповідає за управління фізичною та віртуальною пам'яттю комп'ютера. Це включає управління відображенням віртуальної пам'яті на фізичну, роботу зі сторінками пам'яті та інші аспекти.

*Планування процесів (Process Scheduling):*

Ядро відповідає за планування виконання процесів на процесорі. Це включає вибір процесів для виконання, розподіл часу процесора та призначення пріоритетів.

*Мережеві протоколи (Networking Protocols):*

Ядро містить реалізації різних мережевих протоколів, які дозволяють обмін даними через мережу. Це включає підтримку протоколів TCP/IP, UDP, ICMP та інших.

*Файлові системи (File Systems):*

Ядро надає підтримку різних файлових систем, таких як ext4, btrfs, NTFS та інші. Це дозволяє працювати з файлами та каталогами на зберігаючих пристроях.

*Процеси та потоки (Processes and Threads):*

Ядро керує процесами та потоками в операційній системі, забезпечуючи їх виконання та взаємодію.

## 2. Що таке ядро операційної системи і яке його призначення?

Ядро операційної системи (ОС) є центральною частиною операційної системи, яка відповідає за управління ресурсами комп'ютера та надання середовища для виконання програм.

Призначення ядра операційної системи включає:

*Управління ресурсами:* Ядро контролює доступ до фізичної пам'яті, процесора, введення-виведення та інших ресурсів комп'ютера. Воно розподіляє ресурси між різними програмами та процесами, забезпечуючи їхню взаємодію та виконання.

*Взаємодія з апаратним забезпеченням*: Ядро взаємодіє з апаратним забезпеченням комп'ютера, керуючи взаємодією з пристроями введення-виведення, дисками, мережею та іншими пристроями.

*Забезпечення системних викликів*: Ядро надає інтерфейс для взаємодії програм з операційною системою через системні виклики. Це дозволяє програмам звертатися до різних функцій ОС, таких як робота з файлами, мережею, пам'яттю та іншими ресурсами.

*Управління процесами та потоками*: Ядро відповідає за планування виконання процесів та управління потоками в операційній системі.

*Підтримка мережі та файлових систем*: Ядро надає підтримку мережі та різних файлових систем для забезпечення взаємодії з іншими комп'ютерами та зберігання даних.

## 3. Опишіть загальні частини файлової системи Unix/Linux, архітектуру файлової системи.

Загальні частини файлової системи Unix/Linux:

*Кореневий каталог (Root Directory):*

Кореневий каталог позначений символом "/", він є стартовою точкою для всієї файлової системи. Всі інші файли та каталоги розташовані в межах цього каталогу.

*Каталоги (Directories):*

Файлова система Unix/Linux містить каталоги, які дозволяють організувати файли у структуровану ієрархію. Кожен каталог може містити інші файли та каталоги.

*Файли (Files):*

Файли представляють дані, які можуть бути текстовими, виконуваними програмами, зображеннями тощо. Файли можуть бути організовані у каталогах для зручності управління та доступу.

*Символічні посилання (Symbolic Links):*

Unix/Linux дозволяє створювати символічні посилання, які є посиланнями на інші файли або каталоги. Вони надають зручний спосіб посилання на різні ресурси в межах файлової системи.

*Спеціальні файли (Special Files):*

Файлова система містить спеціальні файли, такі як пристрої (наприклад, /dev/null для відкидання даних) та сокети, які використовуються для міжпроцесного зв'язку.

**Архітектура файлової системи Unix/Linux:**

Файлова система Unix/Linux має ієрархічну структуру, що починається з кореневого каталогу "/" і включає в себе різні каталоги та файли. Основні аспекти архітектури файлової системи включають:

* *Кореневий каталог* ("/"): Це початковий пункт в ієрархії файлової системи*.*
* *Каталоги та підкаталоги*: Файлова система містить каталоги, які можуть містити інші каталоги та файли.
* *Файли*: Дані в файловій системі представлені у вигляді файлів, які можуть мати різний тип та призначення.
* *Символічні посилання*: Unix/Linux дозволяє створювати символічні посилання на файли та каталоги для зручності управління ресурсами.
* *Спеціальні файли*: Файлова система містить спеціальні файли, такі як пристрої та сокети, які надають доступ до апаратного забезпечення та мережевих можливостей.

## 4. У чому різниця між RedHat Linux і Debian Linux?

RedHat Linux та Debian Linux - це обидва популярні дистрибутиви операційної системи Linux, але вони мають деякі важливі відмінності. Ось деякі з них:

*RedHat Linux:*

RedHat Linux базується на комерційній підтримці та спрямований на корпоративний сектор.

Використовує систему керування пакунками RPM (Red Hat Package Manager).

У RedHat зазвичай використовується YUM (Yellowdog Updater, Modified) для керування пакунками.

Один з відомих продуктів RedHat - Red Hat Enterprise Linux (RHEL), який надається з підтримкою та гарантіями.

*Debian Linux:*

Debian є безкоштовним дистрибутивом з відкритим вихідним кодом та спрямований на вільне програмне забезпечення.

Використовує систему керування пакунками dpkg (Debian Package).

Для керування пакунками використовується APT (Advanced Package Tool).

Debian має більшу кількість пакунків у своєму репозиторії порівняно з RedHat.

## 5. У чому різниця між /proc та /sys в операційних системах Linux?

Це дві спеціальні файлові системи в операційній системі Linux, які надають доступ до системної інформації та параметрів ядра. Ось деякі відмінності між ними:

|  |  |
| --- | --- |
| /proc | /sys |
| Містить інформацію про процеси та ресурси системи у вигляді текстових файлів та каталогів. | Містить ієрархію каталогів та файлів, що представляє параметри та налаштування пристроїв та драйверів ядра. |
| Надає доступ до поточної інформації про процеси та ресурси системи в реальному часі. | Дозволяє ядру та пристроям взаємодіяти через спеціальні файли та каталоги. |
| Можна зчитувати та іноді писати дані для зміни параметрів ядра, таких як відстеження процесора та пам'яті. | Надає інтерфейс для зчитування та налаштування параметрів пристроїв, таких як мережеві інтерфейси та блочні пристрої. |
| Містить файли з інформацією про процесори (/proc/cpuinfo), пам'ять (/proc/meminfo) та інші. | Містить інформацію про мережеві інтерфейси (/sys/class/net) та блочні пристрої (/sys/block). |

Якщо /proc надає інформацію про процеси та ресурси системи, то /sys надає доступ до параметрів та налаштувань ядра та пристроїв. Обидва ці шляхи використовуються для взаємодії з операційною системою та її компонентами, проте мають різні призначення та структуру.

## 6. Ситуація: показує, що на диску зайнято 50% місця, а створити файл навіть під root користувачем не можемо. У чому проблема?

*Втрата фізичного доступу до файлової системи,* якщо підключення ведеться за допомогою shh або іншим шляхом.

*Обмеження прав доступу*: Навіть якщо ви користуєтесь обліковим записом користувача root, можуть існувати обмеження на рівні файлової системи або прав доступу.

*Проблеми з файловою системою*: Іншою можливою причиною може бути проблема з файловою системою, така як пошкодження або помилки в системі файлів, які перешкоджають створенню нових файлів.

Для вирішення цієї проблеми, може знадобитися:

* Перевірити наявність доступу до диска і вільне місце на ньому.
* Переконатися, що у вас є права доступу до директорії, в якій ви намагаєтеся створити файл.
* Перевірити цілісність файлової системи та виконати необхідні дії для виправлення можливих проблем.

## 7. Що може створювати високе навантаження на CPU (процеси застосунків споживають дуже мало ресурсів CPU)?

*Мультизадачність*: Якщо ви виконуєте багато програм одночасно, то це може призвести до великого навантаження на CPU, навіть якщо окремі процеси застосунків не споживають багато ресурсів.

*Фонові процеси*: Деякі застосунки можуть запускати фонові процеси, які використовують ресурси CPU, навіть якщо основні процеси застосунку не активні.

*Погано оптимізований код*: Іноді програми можуть мати погано оптимізований код, який призводить до великого навантаження на CPU при виконанні навіть малих завдань.

*Віруси та шкідливі програми*: Шкідливі програми можуть використовувати ресурси CPU без вашого дозволу, що може призвести до високого навантаження.

*Оновлення та фонові завдання ОС*: Операційні системи можуть виконувати фонові завдання, такі як оновлення або планові завдання обслуговування, які можуть спричиняти велике навантаження на CPU.

Якщо ви стикаєтеся з високим навантаженням на CPU, вам може бути корисно використовувати системні засоби моніторингу, щоб виявити, які процеси споживають найбільше ресурсів, а також виконати антивірусну перевірку для виявлення можливих шкідливих програм.

## 8. Ми видалили файл, який відкрив застосунок. Як нам його відновити?

В операційній системі Linux можна видалити файл, який відкрив застосунок. Це може статися через команду видалення файлу, яку ви можете випадково ввести, або через видалення файлу в графічному інтерфейсі операційної системи.

Якщо ви випадково видалили файл, який відкрив застосунок, то вам може знадобитися відновити цей файл, якщо він був важливим. Є кілька можливих шляхів для його відновлення:

* *Кошик*: Перевірте кошик на випадок, якщо файл може бути відновлений з нього. Якщо видалення відбулося нещодавно, файл може знаходитися в кошику.
* *Backup*: Якщо у вас є резервна копія системи або регулярні резервні копії файлів, ви можете спробувати відновити файл з резервної копії.
* *Data Recovery Tools*: Існують спеціалізовані програми для відновлення видалених файлів в операційній системі Linux, такі як TestDisk, PhotoRec тощо. Вони можуть допомогти відновити видалений файл.

## 9. Як в Linux знайти PID процесу, його стартові параметри?

Щоб знайти PID процесу та його стартові параметри в Linux, ви можете скористатися командним рядком та деякими утилітами.

**Знаходження PID процесу:**

Використовуйте команду ps aux | grep для виведення усіх процесів та їх PID. Наприклад,

$ ps aux | grep nginx

виведе інформацію про процеси, пов'язані з веб-сервером Nginx.

Також, можна скористатися командою pgrep для знаходження PID процесу за його назвою. Наприклад,

$ pgrep nginx

поверне PID процесів Nginx.

**Стартові параметри процесу:**

Щоб отримати інформацію про стартові параметри процесу, ви можете використати файл /proc/<PID>/cmdline. Наприклад, для процесу з PID 12345 ви можете переглянути його стартові параметри за допомогою команди

$ cat /proc/12345/cmdline.

Стартові параметри процесу в Linux відображають усю необхідну інформацію про запуск процесу, яка може бути корисною для адміністрування та налагодження системи.

**Стартові параметри процесу** - це дані, які визначають, як процес був запущений або ініційований. Ці параметри включають в себе інформацію про команду, яка викликала процес, аргументи командного рядка, середовище виконання та інші параметри, що впливають на його роботу.

Коли процес запускається в Linux, йому можуть бути передані додаткові параметри через командний рядок.

Стартові параметри процесу можуть бути корисні для налагодження, аналізу та контролю за роботою програм. Наприклад, вони дозволяють розібратися, як саме процес був запущений, які аргументи були передані та які змінні середовища впливають на його роботу.

У Linux стартові параметри процесу зазвичай зберігаються у спеціальній віртуальній файловій системі /proc. Кожен процес має свій власний каталог у цій файловій системі, де можна знайти важливі дані про його виконання, включаючи стартові параметри.

Стартові параметри процесу зберігаються у файлі /proc/<PID>/cmdline, де <PID> - це ідентифікатор процесу (PID). Вміст цього файлу представляє собою рядок символів, що містить команду запуску процесу та його аргументи командного рядка.

## 10. Як в Linux перевірити, чи відкритий порт на віддаленому хості, локальному хості?

У Linux можна перевірити, чи відкритий порт на віддаленому хості та локальному хості за допомогою різних утиліт командного рядка. Ось декілька способів зробити це:

*Перевірка відкритих портів на віддаленому хості*

* Використання утиліти telnet: Ви можете використати команду  
   $ telnet <адреса\_хоста> <порт>  
  для спроби підключення до віддаленого хоста через певний порт. Наприклад, telnet example.com 80 спробує підключитися до хоста example.com через порт 80.
* Використання утиліти nc (netcat): Команда  
   $ nc -zv <адреса\_хоста> <порт>   
  також дозволяє перевірити доступність порту на віддаленому хості. Наприклад,  
   $ nc -zv example.com 80  
  перевірить доступність порту 80 на хості example.com.

*Перевірка відкритих портів на локальному хості*

* Використання утиліти netstat: Ви можете скористатися командою  
   $ netstat -tuln  
  для перегляду списку відкритих портів на локальному хості. Буде виведено список портів, які відкриті для прослуховування на вашому комп'ютері.
* Використання утиліти ss: Команда  
   $ ss -tuln  
  також надасть інформацію про відкриті порти на локальному хості.

## 11. Як в Linux шукати файл за його вмістом?

У Linux для пошуку файлу за його вмістом можна використовувати різні утиліти командного рядка. Ось декілька з них:

Утиліта **grep**: Команда grep дозволяє здійснювати пошук тексту в файлі. Наприклад, для пошуку усіх рядків, які містять певне слово у файлі, ви можете використати команду grep 'шуканий\_текст' шлях\_до\_файлу.

Утиліта **ack** або **ack-grep**: Ця утиліта пропонує більш продуктивний та зручний пошук за замовчуванням. Вона дозволяє швидко шукати вміст файлів за допомогою простих команд.

Утиліта **ag** або **silversearcher-ag**: Ця утиліта також добре підходить для пошуку вмісту файлів у каталозі. Вона відома своєю швидкістю та можливістю використання регулярних виразів.

Утиліта **find** з опцією **-exec**: Ви можете також скористатися командою find для пошуку файлів за їх вмістом та обробки результатів за допомогою опції -exec.

## 12. Що таке SSH?

SSH у Linux - це протокол та програмне забезпечення для безпечного з'єднання з віддаленим комп'ютером чи сервером через захищену мережу. SSH розшифровується як "Secure Shell".

Основні особливості SSH включають:

* *Шифрування*: Усі дані, що передаються через SSH, шифруються, що забезпечує конфіденційність інформації під час передачі.
* *Автентифікація*: SSH використовує механізми автентифікації, такі як паролі, ключі SSH або інші методи, щоб переконатися, що тільки правильні користувачі мають доступ до віддаленого сервера.
* *Тунельні з'єднання:* SSH дозволяє створювати тунельні з'єднання для безпечної передачі даних між вузлами мережі.
* *Виконання віддалених команд*: SSH дозволяє виконувати команди на віддаленому сервері через командний рядок.

SSH є важливим інструментом для адміністраторів систем та користувачів, які потребують безпечного та захищеного доступу до віддалених ресурсів.

## 13. Як в Linux організувати доступ на сервер без пароля або з певних хостів?

У Linux можна організувати доступ на сервер без пароля або з певних хостів за допомогою ключів SSH та файлу ~/.ssh/authorized\_keys. Ось кроки для цього:

*Створення ключів SSH*: Спочатку вам потрібно створити ключі SSH на клієнтському комп'ютері за допомогою команди ssh-keygen. Ця команда створить ключі RSA або ECDSA (еліптична крива криптографія) для вас.

*Передача ключа на сервер*: Після створення ключів ви повинні передати громадський ключ (зазвичай /.ssh/id\_rsa.pub або /.ssh/id\_ecdsa.pub) на сервер за допомогою команди

$ ssh-copy-id user@server

Ця команда автоматично додасть ваш громадський ключ до файлу ~/.ssh/authorized\_keys на віддаленому сервері.

*Налаштування доступу без пароля*: Після передачі ключа на сервер ви зможете увійти на сервер без пароля, використовуючи ваш приватний ключ.

*Обмеження доступу за IP-адресою*: Щоб обмежити доступ до сервера з певних хостів, ви можете внести відповідні налаштування в файл sshd\_config на сервері. Наприклад, ви можете сконфігурувати файрвол або SSH-сервер так, щоб дозволяти з'єднання лише з певних IP-адрес чи діапазонів IP-адрес.

Ці кроки допоможуть вам організувати доступ на сервер без пароля або з певних хостів за допомогою ключів SSH та обмежень доступу за IP-адресою.

## 14. Як в Linux обмежити доступні для виконання команди?

В Linux ви можете обмежити доступні для виконання команди, використовуючи такі методи:

Використання **sudo**: Ви можете налаштувати файл *sudoers*, щоб обмежити доступ користувачів до певних команд або програм. Це можна зробити, додаючи відповідні правила до файлу *sudoers*. Наприклад, ви можете дозволити користувачеві виконувати певну команду з підвищеними привілеями, без введення пароля.

Використання **AppArmor** або **SELinux**: Це системи контролю доступу, які дозволяють обмежувати доступ до файлів, каталогів та програм на рівні ядра операційної системи. Вони забезпечують механізми для визначення та обмеження виконання певних програм та команд.

Використання обмежень **shell**: Ви можете використовувати обмеження оболонки (наприклад, *rbash* або restricted shell) для обмеження доступних команд для користувача. Це дозволить вам обмежити користувача від виконання певних команд та обмежити його доступ до певних функцій оболонки.

Використання **контейнерів**: Використання технологій контейнеризації, таких як Docker або LXC, дозволяє вам ізолювати виконання програм та команд в окремому середовищі. Ви можете налаштувати контейнер таким чином, щоб обмежити доступні для виконання команди всередині контейнера.

Ці методи дозволяють обмежити доступні для виконання команди в Linux залежно від ваших потреб та вимог безпеки.

## 15. Як перевірити спожиті ресурси під час сеансу SSH?

Ось деякі корисні команди:

* *top*: Використання команди top під час сеансу SSH дозволяє переглянути список процесів, які використовують ресурси процесора та пам'яті. Для виходу з режиму top, натисніть клавішу q.
* *htop*: Команда htop надає більш зручний та інтерактивний спосіб перегляду споживаних ресурсів. Вона показує список процесів у вигляді зручної таблиці та дозволяє взаємодіяти з ними, наприклад, завершувати процеси або змінювати їх пріоритет.
* *free*: Команда free дозволяє переглянути використання пам'яті та пам'яті, доступної для використання.
* *vmstat*: Використання команди vmstat дозволяє отримати інформацію про використання віртуальної пам'яті, процесора та введено-виведеної діяльності.
* *sar*: Команда sar надає статистику використання ресурсів на сервері, таку як центральний процесор, пам'ять та мережа.

## 16. Що означає дозвіл на файл 755 в операційній системі Linux?

Дозвіл на файл 755 означає, що власник файлу має повний доступ (читання, запис та виконання), а група користувачів та інші користувачі мають дозвіл на читання та виконання файлу, але не на його змінення.

У Linux структура цифрового коду file permissions представлена трьома числами, які відповідають за права доступу для власника, групи та інших користувачів:

Власник (Owner): Дозволи власника файлу.

Група (Group): Дозволи групи, до якої належить файл.

Інші користувачі (Others): Дозволи для всіх інших користувачів.

Кожне з цих чисел складається з трьох бітів, які представляють дозволи на читання, запис та виконання відповідно. Ось як ця структура представляється:

Читання (read) = 4

Запис (write) = 2

Виконання (execute) = 1

## 17. Що таке SELinux і навіщо він потрібен?

SELinux (Security-Enhanced Linux) - це обов'язкова система контролю доступу в Linux, яка надає додаткові рівні безпеки для операційної системи на рівні ядра. Основна мета SELinux полягає в обмеженні доступу процесів до ресурсів в операційній системі, щоб запобігти зловживанням та атакам.

**Навіщо потрібен SELinux:**

*Захист від вразливостей*: SELinux дозволяє обмежити доступ процесів до ресурсів та системних об'єктів, що допомагає запобігти використанню вразливостей в програмному забезпеченні.

*Мінімізація ризиків*: Шляхом обмеження доступу до ресурсів, SELinux допомагає мінімізувати можливі ризики, пов'язані з несанкціонованим доступом та атаками на систему.

*Захист конфіденційності даних*: Шляхом обмеження доступу до файлів та ресурсів, SELinux допомагає захистити конфіденційні дані в операційній системі.

*Відокремлення привілегій*: SELinux допомагає відокремити рівні привілегій та обмежити можливості атак на систему.

*Забезпечення відповідності*: Для деяких організацій SELinux допомагає забезпечити відповідність з вимогами безпеки та регуляторними стандартами.

У цілому, SELinux є потужним інструментом для забезпечення безпеки операційних систем Linux шляхом обмеження доступу до ресурсів та захисту від потенційних загроз безпеки.

## 18. Як визначити PCI-пристрій у системі, наприклад RAID controller?

Ось декілька способів визначення PCI-пристроїв:

**lspci**: Команда *lspci* дозволяє вивести список всіх PCI-пристроїв, підключених до вашої системи. Ви можете виконати цю команду в терміналі, і вона покаже список PCI-пристроїв, включаючи контролери RAID, які можуть бути встановлені у вашій системі.

$ lspci | grep -i raid

**lshw**: Команда *lshw* виводить докладну інформацію про апаратне забезпечення вашої системи, включаючи PCI-пристрої. Ви можете використати цю команду, щоб отримати більш докладну інформацію про контролер RAID та інші PCI-пристрої.

$ lshw -class storage

**dmidecode**: Ще одним способом визначити PCI-пристрої у вашій системі є використання команди dmidecode, яка надасть інформацію про апаратне забезпечення, включаючи PCI-пристрої.

$ dmidecode -t slot

## 19. Як в Linux перейменувати пристрій, наприклад мережеву карту чи диск?

Ось декілька способів це зробити:

*Перейменування мережевої карти:*

* Використання **udev**: Можна використовувати *udev* для перейменування мережевих карт за допомогою правил. Спочатку вам потрібно з'ясувати, яка мережева карта вам потрібна. Потім можна створити правило udev для перейменування мережевої карти.
* Використання **ifrename**: Іншим способом є використання утиліти *ifrename*, яка дозволяє перейменовувати мережеві карти на основі правил, визначених у конфігураційному файлі.

*Перейменування дисків:*

* Використання утиліти **mv**: Якщо вам потрібно перейменувати диск або його розділ, ви можете використати утиліту mv для перейменування точки монтування або шляху до диску. Однак, це може бути складніше у випадку, коли диск використовується.
* Використання утиліти **e2label**: Якщо ви працюєте з файловою системою ext2, ext3 або ext4, можна використовувати утиліту *e2label* для зміни мітки диску, що є одним зі способів ідентифікації диску.

Перед внесенням будь-яких змін у конфігурацію мережевих карт або дисків рекомендується ретельно продумати та зробити резервну копію важливих даних, оскільки ці зміни можуть вплинути на функціонування вашої системи.

## 20. Що таке LVM? Які знаєте приклади використання?

LVM (Logical Volume Manager) - це програмне забезпечення для управління логічними томами на Linux. Воно дозволяє об'єднувати фізичні носії даних, такі як жорсткі диски або розділи, в один або кілька логічних томів, які можна легко змінювати за рахунок розширення або зменшення їх розміру.

*Приклади використання LVM включають:*

* Розширення та зменшення томів без перетворення файлової системи: LVM дозволяє змінювати розмір логічних томів без необхідності перетворення файлової системи. Це дозволяє легко розширювати простір для зберігання даних без необхідності резервного копіювання та відновлення даних.
* Створення знімків: LVM дозволяє створювати знімки логічних томів, що дозволяє зберегти стан даних на певний момент часу. Це може бути корисним для створення резервних копій або тестування оновлень програмного забезпечення.
* Об'єднання різних фізичних носіїв: LVM дозволяє об'єднувати різні фізичні носії даних в один логічний том. Це може бути корисним для створення великих збережень даних з використанням декількох жорстких дисків.
* Управління простором для зберігання даних на льоту: LVM дозволяє легко розподіляти, об'єднувати та перерозподіляти простір для зберігання даних на льоту, що дозволяє ефективно використовувати фізичні ресурси та забезпечувати гнучкість управління простором для зберігання даних.

LVM є потужним інструментом для управління простором для зберігання даних на Linux і забезпечує багато можливостей для гнучкого та ефективного управління даними.

## 21. Що таке root reserved space?

Root reserved space, або reserved space for root, це частина простору на файловій системі, яка залишається вільною для використання тільки адміністратором (root) системи. Ця практика застосовується для забезпечення безпеки та стабільності системи, навіть коли вона заповнюється даними.

Reserved space дозволяє системі залишати додатковий простір для важливих операцій, таких як запис журналів, виконання системних операцій, запобігання фрагментації файлової системи та уникнення ситуацій, коли система може перестати працювати через повний розмір файлової системи.

Зазвичай, reserved space складається з певного відсотка вільного місця на файловій системі, який призначений тільки для користувача root. Наприклад, для захищених файлових систем, таких як ext4, за замовчуванням reserved space становить 5%. Це означає, що тільки користувач root може використовувати цей простір, навіть якщо інші користувачі вже вичерпали весь доступний простір.

Reserved space грає важливу роль у забезпеченні стабільності та безпеки операцій з даними на системі.

## 22. Що таке “exit code” та як його дізнатися?

Exit code (код виходу) - це значення змінної оточення, яке повертається з програми або процесу після його завершення. Це числове значення вказує на статус завершення процесу і може містити інформацію про те, чи відбулася помилка під час виконання програми.

Щоб дізнатися exit code для попередньо виконаної команди в UNIX-подібних системах, ви можете використати змінну $? після виконання команди. Наприклад, в терміналі Linux ви можете ввести:

$ echo $?

Це виведе числове значення, яке представляє код виходу для попередньо виконаної команди. Зазвичай значення 0 вказує на успішне завершення, а будь-яке інше значення може вказувати на наявність помилки або проблеми під час виконання команди.

Exit code є засобом для розуміння статусу виконання програм та автоматизації операцій в скриптах та інших автоматизованих процесах.

## 23. Чому вивід команди df -h показує, що на диску зайнято мало місця, але система не дає записати файл із повідомленням “no space left on device”?

Вивід команди df -h показує розподіл простору на диску та його використання. Якщо ця команда показує, що на диску зайнято мало місця, але система не дозволяє записати файл із повідомленням "no space left on device", це може мати кілька можливих причин:

*Резервований простір для root*: У багатьох файлових системах резервується певна кількість простору для використання root-користувачем. Це означає, що навіть якщо для звичайних користувачів простір виглядає вільним, root-користувач може бачити бракуючий простір. Це може бути причиною повідомлення про "no space left on device".

*Фрагментація файлової системи*: Іноді файлова система може бути зайнята багатьма невеликими фрагментами, які утворюють "дірки" у вільному просторі. Хоча загальний обсяг вільного місця може бути значним, він може бути розташований у недоступних для запису ділянках файлової системи.

*Системні процеси, що використовують простір*: Іноді системні процеси або служби можуть використовувати простір на диску, що не відображається в звичайних вимірах вільного місця.

Щоб дізнатися точну причину, може бути корисним використання інших команд, таких як **du** для аналізу використання простору на диску та **df -i** для перевірки використання інодів. Також, перевірте журнали системи для отримання більш детальної інформації про помилки запису файлів.

## 24. У чому різниця між command1 & command2 та command1 && command2, а також command1 && command2 || command3?

*Команда з одним амперсандом (command1 & command2):*

Команда command1 виконується в фоновому режимі, тобто у фоновому процесі.

Після запуску command1, виконується command2 незалежно від того, чи завершилася command1 чи ні.

Це дозволяє виконувати команди паралельно, без очікування завершення першої команди.

*Команда з двома амперсандами, тобто логічний оператор “і” (command1 && command2):*

Команда command1 виконується, і тільки якщо вона завершується успішно (з вихідним кодом 0), тоді виконується command2.

Якщо command1 завершується з помилкою (з вихідним кодом, відмінним від 0), то command2 не виконується.

*Логічний оператор “і”, потім логічний оператор* ***АБО*** *(command1 && command2 || command3):*

Команда command1 виконується, і тільки якщо вона завершується успішно (з вихідним кодом 0), тоді виконується command2. Якщо command1 завершується з помилкою, command2 не виконується, а виконується command3.

Тобто, command3 виконується лише в тому випадку, коли command1 завершується з помилкою.

## 25. З мережі різко зріс вихідний трафік на 25-й порт. Як, маючи доступ на gateway, виявити шкідника з внутрішньої мережі?

Зазвичай порт 25 використовується для протоколу SMTP (Simple Mail Transfer Protocol), який є стандартним протоколом для надсилання електронної пошти через Інтернет.

Для його перевірки ви можете використати кілька методів та інструментів:

*Моніторинг мережевого трафіку:*

Використовуйте програми для моніторингу мережі, такі як Wireshark або tcpdump, для аналізу трафіку на 25-му порті. Вони дозволять вам переглядати пакети, які проходять через мережу, і виявляти незвичайну активність або незвичайний трафік на цьому порті.

*Журнали мережевої активності:*

Перевірте журнали мережевої активності на gateway, щоб знайти незвичайну активність на 25-му порті. Це може допомогти виявити IP-адреси або пристрої, які генерують надмірний трафік.

*Використання інструментів моніторингу мережі:*

Встановіть інструменти моніторингу мережі (наприклад, ntop, iftop) на gateway, щоб відстежити, які пристрої відправляють або отримують незвичайний трафік через 25-й порт.

*Перевірка систем безпеки:*

Перевірте системи безпеки на внутрішній мережі, такі як файервол та системи виявлення вторгнень (IDS/IPS), щоб знайти аномальну активність або спроби вторгнення через 25-й порт.

*Перевірка наявності вразливостей:*

Перевірте всі пристрої внутрішньої мережі на наявність вразливостей, які можуть бути використані для генерації надмірного трафіку.

## 26. Як виконати тонку настройку параметрів Linux Kernel?

Для тонкої настройки параметрів Linux Kernel ви можете використати кілька підходів та інструментів:

*Sysctl:*

Використовуйте утиліту sysctl для зміни параметрів ядра в реальному часі та для збереження цих змін після перезавантаження. Наприклад, ви можете встановити значення параметра, використовуючи команду sysctl -w, та зберегти ці зміни в файлі конфігурації /etc/sysctl.conf.

*Файли параметрів ядра в /proc:*

Багато параметрів ядра можна налаштувати, змінивши значення відповідних файлів у каталозі /proc/sys/. Наприклад, ви можете змінити максимальну кількість відкритих файлів шляхом зміни значення в файлі /proc/sys/fs/file-max.

*Файли конфігурації ядра:*

В більшості випадків параметри ядра можна налаштувати за допомогою файлів конфігурації, таких як /etc/sysctl.conf або відповідних файлів у каталозі /etc/sysctl.d/. Ви можете встановлювати значення параметрів та вказувати їх в цих конфігураційних файлах.

*Компіляція ядра:*

Якщо вам потрібно внести глибокі зміни в параметри ядра, ви можете змінити параметри під час компіляції ядра Linux. Це дасть вам більший контроль над параметрами ядра та можливість внести значні зміни.

Перед внесенням будь-яких змін у параметри ядра рекомендується ретельно розглянути наслідки цих змін та впевнитися, що вони відповідають вашим потребам та не порушують стабільність системи.

## 27. Що таке ulimits в операційній системі Linux?

Ulimits (або "user limits") в операційній системі Linux використовуються для обмеження ресурсів, доступних для користувачів та процесів. Ці обмеження можуть бути встановлені адміністраторами системи для забезпечення безпеки, стабільності та ефективності використання ресурсів.

Нижче наведено деякі ресурси, які можуть бути обмежені за допомогою ulimits:

*Кількість відкритих файлів*: Ulimits можуть бути використані для обмеження кількості відкритих файлів для кожного користувача або процесу.

*Кількість створених процесів*: Адміністратори можуть встановлювати обмеження на кількість процесів, які може створити користувач або процес.

*Обсяг вживаної пам'яті*: Можливо, встановити обмеження на обсяг пам'яті, доступний для користувачів або процесів.

*Час виконання процесів*: Ulimits можуть бути використані для обмеження максимального часу виконання для процесів.

Ці обмеження можуть бути встановлені для конкретних користувачів або груп користувачів, а також для конкретних процесів. Вони можуть бути корисні для управління ресурсами системи, запобігання перевантаженням та забезпечення стабільності в роботі операційної системи.

## 28. У чому різниця між символічними та hard links?

У Linux символічні посилання (symbolic links або симлінки) - це спеціальний тип файлів, який дозволяє створювати посилання на інші файли або каталоги. Ось деякі ключові особливості символічних посилань в Linux:

* Посилання на інші файли або каталоги: Символічні посилання дозволяють створювати посилання на інші файли або каталоги в файловій системі. Це дає можливість створювати віртуальні шляхи до файлів чи каталогів, які можуть бути в іншому місці файлової системи.
* Відмінність від жорстких посилань: У відмінність від жорстких посилань, символічні посилання містять шлях до цільового файлу, тоді як жорсткі посилання містять пряме посилання на *inode* файлу.
* Можливість посилання на файли з різних файлових систем: Символічні посилання можуть посилатися на файли, які знаходяться в різних файлових системах, що дозволяє зручно організовувати структуру файлів.
* Легка зміна або видалення: symbolic links легко змінювати або видаляти, оскільки вони представляють собою просто посилання на цільовий файл.
* Використання команди ln: Для створення символічного посилання використовується команда  
   $ ln -s source\_file target\_link  
  де source\_file - це цільовий файл або каталог, а target\_link - це ім'я символічного посилання.

## 29. Що таке фрагментація файлових систем ext3 та ext4?

Фрагментація файлової системи відноситься до розподілу файлів на дискі у непослідовному порядку, що може виникати при додаванні, видаленні та зміні розміру файлів. Рівень фрагментації може впливати на швидкодію зчитування та запису, оскільки операційна система повинна виконати додаткові операції для знаходження та зчитування різних фрагментів файлу.

У випадку файлових систем ext3 та ext4 в Linux, фрагментація може виникати при додаванні, видаленні та зміні розміру файлів. Однак, варто відзначити, що файлові системи ext4 мають покращену систему управління блоками та алокації місця, що зменшує ймовірність фрагментації порівняно з ext3.

Ось деякі ключові відмінності між фрагментацією в файлових системах ext3 та ext4:

ext3:

Файли, що змінюють розмір або зміст, можуть стати фрагментованими внаслідок видалення та додавання блоків даних.

Можлива зміна продуктивності під час роботи з фрагментованими файлами.

ext4:

Механізми алокації місця та управління блоками покращені, що зменшує ймовірність фрагментації. Файли менше схильні до фрагментації порівняно з ext3.

Загалом, файлова система ext4 пропонує покращену продуктивність та зменшену фрагментацію порівняно з ext3, що робить її більш ефективною для сучасних операційних систем Linux.

## 30. Навіщо файлові системи ext\* операційної системі Linux резервують 5% місця?

Файлові системи ext2, ext3 та ext4 в операційній системі Linux резервують 5% місця для спеціального використання системою, і це має кілька важливих причин:

*Запобігання фрагментації файлів*: Резервування 5% місця дозволяє файловій системі залишати додатковий вільний простір, який може бути використаний для уникнення фрагментації файлів та забезпечення швидкодії операцій в майбутньому. Це особливо важливо на серверах та в системах, де робота з файлами великого розміру є постійною.

*Зменшення фрагментації метаданих*: Резервування простору допомагає уникнути фрагментації метаданих, таких як inode tables та інших системних ресурсів, що допомагає забезпечити стабільну роботу файлової системи.

*Запобігання заповненню диска*: Резервування 5% місця надає буферне місце для важливих системних операцій в умовах, коли диск може бути заповнений до максимуму. Це дозволяє системі продовжувати нормальне функціонування і вести логи, навіть коли диск майже заповнений.

## 31. Як збільшити розмір файлової системи?

Щоб збільшити розмір файлової системи в операційній системі Linux, можна виконати наступні кроки:

Збільшіть розмір розділу: Якщо ви використовуєте класичний розділ на диску, спочатку вам потрібно збільшити його розмір. Це можна зробити за допомогою інструментів, таких як fdisk або parted. Важливо забезпечити, щоб збільшення розділу не порушило існуючих даних.

Оновіть розмір файлової системи: Після збільшення розділу вам потрібно оновити розмір самої файлової системи. Наприклад, якщо ви використовуєте ext4, ви можете використати команду resize2fs для зміни розміру файлової системи на новий розмір розділу.

Перевірте та перезавантажте: Після зміни розміру файлової системи важливо перевірити її на наявність помилок за допомогою команди e2fsck. Після цього можна перезавантажити систему для застосування змін.

Ці кроки можуть відрізнятися в залежності від конкретної конфігурації вашої системи та використовуваної файлової системи, тому рекомендується виконати резервне копіювання важливих даних та документацію перед внесенням будь-яких змін у розмір файлової системи.

## 32. Чи можемо ми зменшити розмір файлової системи?

Так, у операційній системі Linux можна зменшити розмір файлової системи, але цей процес потребує обережності та вимагає деяких кроків для безпечного зменшення розміру.

Ось кілька кроків, які можна виконати для зменшення розміру файлової системи:

Резервне копіювання даних: Перш ніж розпочати зменшення розміру файлової системи, важливо зробити резервне копіювання всіх важливих даних. Це може запобігти втраті даних в разі непередбачуваних проблем під час процесу зменшення розміру.

Зменшення розміру файлової системи: Використовуючи відповідний інструмент, такий як resize2fs для файлових систем ext\*, можна зменшити розмір самої файлової системи до нового значення.

Зменшення розділу на диску: Після зменшення розміру файлової системи, можна зменшити розділ на диску, за допомогою інструментів, таких як fdisk або parted.

Перевірка та перезавантаження: Після зменшення розділу важливо перевірити файлову систему на наявність помилок за допомогою e2fsck, а потім перезавантажити систему для застосування змін.

Рекомендується докладно ознайомитися з документацією та виконати резервне копіювання даних перед початком цього процесу.

## 33. Що таке chroot і навіщо він потрібний?

Chroot (скорочено від "change root") - це системний виклик у багатьох операційних системах, включаючи Linux, який дозволяє змінити кореневий каталог для поточного процесу та всіх його дочірніх процесів. Цей механізм створює віртуальне середовище, обмежуючи доступ до файлової системи тільки до визначеного каталогу.

Ось декілька важливих використань chroot:

І*золяція процесів*: chroot дозволяє ізолювати процеси від інших частин файлової системи. Це може бути корисним для запуску додатків в обмеженому середовищі, яке не має прямого доступу до основної файлової системи.

*Відновлення системи*: chroot може бути використаний для відновлення пошкоджених або недоступних систем, надаючи змогу запустити операційну систему з іншого кореневого каталогу.

*Тестування та розробка*: Розробники часто використовують chroot для тестування програм в контрольованому середовищі, щоб уникнути впливу на основну систему.

*Безпека*: chroot використовується для обмеження доступу до файлової системи, що може знизити ризики вразливостей безпеки в деяких сценаріях.

chroot використовується для створення ізольованого середовища в операційній системі, де процеси можуть працювати в обмеженому каталозі з метою безпеки, тестування та відновлення системи.

## 34. У нас є Linux box з 2 Гб оперативної пам’яті та Java-застосунок, який намагається виділити 4 Гб під час запуску. Чи вдасться це?

Ні, у нас не вийде виділити 4 Гб оперативної пам'яті для Java-застосунку на Linux box з 2 Гб оперативної пам'яті. Ваша система не зможе виділити більше оперативної пам'яті, ніж та, що фізично доступна.

Якщо спробувати виділити більше пам'яті, ніж доступно, це може призвести до виникнення помилки або навіть до аварійного завершення роботи програми. Тому вам слід або зменшити потребу застосунку в оперативній пам'яті, або розглянути можливість збільшення обсягу оперативної пам'яті на вашій системі.

## 35. Є програма, яка читає файл, який користувач намагається видалити. Чи можна видалити цей файл? Що трапиться, якщо спробувати?

Так, файл можна видалити, навіть якщо його читає програма. У більшості операційних систем, включаючи Linux, видалення файлу, який використовується програмою, можливе. Однак, після видалення файлу програма, яка продовжує його читати, може втратити доступ до його вмісту, що може призвести до непередбачуваної поведінки.

Щоб видалити файл, який використовується програмою, можна скористатися командою rm, проте це може призвести до неправильної роботи програми, яка його читає. Краще спочатку завершити роботу програми, яка використовує файл, а потім видалити файл.

Існують також спеціальні системні виклики, такі як unlink, що дозволяють видаляти файл, який використовується програмою, але це також може призвести до непередбачуваної поведінки програми.

Таким чином, видалення файлу, який використовується програмою, можливе, але це може призвести до проблем з доступом до даних, що використовуються програмою.

## 36. Є програма, яка читає файл, який користувач намагається видалити. Чи можна відновити цей файл? Що трапиться, якщо спробувати?

Якщо файл, який читається програмою, буде видалений, то він не зникне миттєво з пам'яті. Операційна система позначить його як видалений, але фактично вміст файлу залишиться на диску до тих пір, поки його простір не буде перезаписаний іншими даними. Таким чином, у принципі, файл можна відновити, якщо після його видалення не було створено нових файлів або відбулися інші операції, що призвели до перезапису даних на диску.

Для відновлення видаленого файлу можна використати спеціалізовані програми для відновлення даних. Ці програми можуть сканувати носій зберігання і відновлювати видалені файли, якщо їх вміст ще не був перезаписаний. Однак, успішність відновлення файлу залежатиме від того, наскільки багато нових даних було записано на диск після видалення файлу.

## 37. Які механізми створення процесів у Linux ви знаєте?

У Linux існує кілька механізмів створення процесів, які надають різні можливості для управління процесами. Ось кілька з них:

*Fork*: Один з найпоширеніших механізмів створення процесів у Linux. Він полягає в тому, що існуючий процес робить копію самого себе, утворюючи таким чином новий процес-нащадок. Після створення копії відбувається подальше виконання коду в обох процесах.

*Clone*: Цей механізм дозволяє створювати процеси, які можуть мати різні рівні віртуальних ресурсів, таких як пам'ять, файлові дескриптори і т. д. Він дозволяє створювати "легкі" процеси (або потоки), які можуть ділити деякі ресурси з батьківським процесом.

*vfork*: Цей механізм схожий на fork, але він гарантує, що дочірній процес не викликається до завершення виконання або виклику exec або exit. Це корисно для оптимізації виконання нових програм в контексті дочірнього процесу.

*exec*: Цей механізм використовується для заміни поточного образу процесу новим образом, який завантажується з виконуваного файлу. В результаті виклику exec старий образ процесу замінюється новим, і виконуваний код починає виконуватися з початку.

## 38. Порівняйте systemd та init system для операційної системи Linux.

Системи ініціалізації systemd та init є двома різними підходами до керування процесами та ініціалізації операційної системи Linux. Ось порівняння цих двох систем:

**init system:**

* *Простота*: Система ініціалізації init, така як SysVinit, працює за допомогою скриптів ініціалізації, які виконуються послідовно для запуску системи. Це простий та прямолінійний підхід.
* *Обмежені можливості*: init має обмежені можливості управління процесами та ресурсами. Він може виконувати лише послідовний запуск служб без багатьох розширених можливостей управління.

**systemd:**

* *Розширені можливості*: systemd надає додаткові можливості управління процесами, ресурсами та службами. Воно працює як система керування та ініціалізації, що надає багато функцій, таких як паралельний запуск служб, управління їх залежностями, обмеження ресурсів тощо.
* *Інтеграція з cgroups*: systemd інтегрується з контролерами груп процесів (cgroups) для кращого контролю ресурсів та ізоляції процесів.

У багатьох сучасних дистрибутивах Linux використовується systemd замість старіших систем ініціалізації, оскільки він надає більше можливостей та кращу інтеграцію з сучасними функціями операційних систем.

## 39. У нас є директорія з великою кількістю файлів, і ми хочем видалити всі файли з іменами, що починаються на A (велика літера). Але команда rm -f A\* видає ‘Argument list too long’. Як можна видалити ці файли?

Щоб видалити файли з іменами, що починаються на A (велика літера), коли ви отримуєте помилку "Argument list too long", можна скористатися командою find разом з командою rm.

Ось приклад команди, яку можна використати:

$ find . -maxdepth 1 -type f -name 'A\*' -exec rm -f {} \;

Ця команда використовує find для пошуку у поточній директорії всіх файлів з іменами, які починаються на A, та використовує команду rm для видалення кожного знайденого файлу.

*The -exec option in the find command requires a semicolon ; or a plus + at the end to properly terminate the command that follows it.*

## 40. Ви починаєте видаляти велику кількість файлів в Linux, використовуючи ‘find . -maxdepth 1 -type f -name 'A\*' -exec rm -f {} \;’ , але кожен виклик rm запитує підтвердження. Як можна автоматизувати цю операцію?

Щоб автоматизувати видалення файлів без підтвердження, можна використати опцію -f для команди rm. Однак, варто бути обережним, оскільки це може призвести до видалення файлів без підтвердження, включаючи важливі файли.

Якщо ви впевнені, що хочете видалити всі файли, що починаються на 'A' в поточній директорії без підтвердження, ви можете модифікувати команду find наступним чином:

$ find . -maxdepth 1 -type f -name 'A\*' -exec rm -f {} +

Замість ; в кінці команди, ви можете використати +, що дозволить видалити файли без підтвердження. Однак опція -f призведе до безповоротного видалення файлів без підтвердження.

* Find . - This command starts the search in the current directory and its subdirectories.
* -maxdepth 1: This option ensures that the search is only conducted in the current directory, without descending into subdirectories.
* -type f - This flag specifies that the search should only consider regular files, not directories or other types of files.
* -name 'A\*' - This part of the command filters the search to only include files whose names start with the letter 'A'.
* -exec rm -f {} + When the files matching the criteria are found, this part of the command executes the rm -f command on them. The rm -f command forcefully removes the files without prompting for confirmation.

*The -exec option in the find command requires a semicolon ; or a plus + at the end to properly terminate the command that follows it.*

# Mddle DevOps - Networks

## 41. Розкажіть про модель OSI. Опишіть функції та призначення кожного рівня.

## 42.Які мережеві топології ви знаєте? Опишіть різницю між ними.

39. Навіщо потрібна IP-адреса, якщо MAC-адреса унікальна? Хіба ми не можемо спілкуватися тільки за MAC-адресою?

40. У чому різниця між концентратором і комутатором L2 у мережах Ethernet?

41. Що таке Vlan і навіщо існує поділ на віртуальні локальні мережі?

42. Який номер порту використовують для PING-комунікації?

43. Що таке сеанс зв’язку? Який алгоритм використовує TCP для доставки?

44. У чому основна відмінність між TCP і UDP?

45. Навіщо нам маршрутизатор за замовчуванням?

46. Як хост вирішує DNS за замовчуванням?

47. Комп’ютер почав отримувати IP-адресу з іншої мережі (є підозра, що в мережі працює інший DHCP-сервер): як його знайти і від’єднати? Які є методи захисту від такої проблеми?

48. Ми будемо мігрувати сайт на нову IP-адресу. Як зробити, щоб користувачі цього практично не помітили?

49. Що таке socket?

50. Як дізнатися, які віддалені хости під’єднуються до хосту через порт 8888? (за допомогою команд і не використовуючи /proc чи /sys).

51. Ми маємо кілька мережевих карт. Як збільшити пропускну спроможність сервера?

52. Як перевірити відкриті порти на віддаленому сервері без команди Netcat чи Nmap Linux?

# Mddle DevOps - Container orchestration

53. У чому переваги Kubernetes як платформи?

54. Що таке control plane та з яких компонентів складається?

55. Які CNI ви використовували та чим вони відрізняються?

56.Чим відрізняється managed Kubernetes від self-deployed?

57.Яким чином можна контролювати розміщення подів у кластері? (taints/tolerations, affinities, topologies etc)

58.Скейлінг кластеру. Cluster autoscaler vs HPA vs VPA? Як зробити zero-downtime node decommission/cluster upgrade? PDB? Lifecycle hooks?

59.Які існують способи для зовнішнього доступу до кластеру? ingress, node port, port-forward тощо.

60.З яким PID запускається процес у контейнері?

61.Що краще використовувати для ізоляції оточення — Vagrant чи Docker?

62.Який інструмент оркестрування контейнерів використовували? (Swarm, Kubernetes, Openshift, Rancher тощо.)

63.Що відбувається в Kubernetes після запуску kubectl (API, ReplicaSet Controller, storage back-end, scheduler, kubelet, worker node, pod)?

64.Яка різниця між pod та контейнером у K8s?

65.Як ми можемо зробити будь-який мікросервіс, який працює на K8s, доступним із зовнішнього середовища?

Віртуалізація і контейнеризація

66.Які типи віртуалізації ви знаєте?

67.Як працює Docker на macOS/Windows?

68.Що таке Docker-image і Docker-контейнер? Як вони між собою пов’язані?

69.Які основні відмінності між контейнерами докерів і віртуальними машинами?

70.Що таке image layer? Яка максимальна кількість layers можлива? Чому треба намагатися мати малу кількість layers? Яка оптимальна кількість?

71.Як у віртуальній машині змінити розмір диска після створення? Що треба зробити з гостьовою ОС?

72.Як у Docker реалізовано обмеження ресурсів?

73.Існує віртуальна машина, до якої втрачено доступ. Як, маючи доступ до її диска, відновити root пароль/SSH-ключ?

74.Оптимізувати Dockerfile, пояснити, що і чому так:

FROM golang

RUN apt install -y pkg1 pkg2 pkgN # Dependencies for app

COPY. .

RUN go build -o app main.go

CMD ./app

75.Що таке IPVS та який функціонал у нього є?

76.Яка структура API у Kubernetes?

77.Що таке operators і для чого вони потрібні?

CI/CD

78.Які стадії мають бути в будь-якому пайплайні (lint, test, build, deploy etc)?

79.Як і де зберігати build artifacts?

80.Що таке артефакт?

81.Є два бренчі: dev і stage. Ми закинули Dockerfile у dev, а потім збілдили у dev і stage. Це буде одним артефактом чи різними?

82.Що ви використовували для автоматизації налаштування Jenkins and GitLab CI?

83.Порівняйте CI інструментів: Jenkins, GitLab CI, AWS Code Pipeline, GCP cloudbuild, GitHub actions, Circle CI.

84.Deployment strategies. Які існують і чим відрізняються (recreate, blue-green, canary etc)?

85.Яким чином реалізувати СI/CD для програми, яка залежить від кількох інших програм?

86.GitOps. В чому його переваги та недоліки?

Clouds and Automation

87.Яка роль і переваги хмарних сервісів для DevOps?

88.Що таке immutable infrastructure? Як досягти? У чому переваги та недоліки? Packer, AMI тощо.

89.Структура Terraform. Як організувати multi-environment project? Terraform workspaces?

90.Найкращі практики з використання багатьох Terraform states.

91.Як організувати доступ розробницькій команді до AWS/GCP/Azure? Role-based access, assume role, SSO.

92.Що таке Terraform provider, module?

93.Як версіонувати Terraform modules?

94.Коли потрібно використовувати local-exec та remote-exec?

95.Що таке golden image та як його створити?

Monitoring/Logging

96.Як моніторинг допомагає підтримувати всю архітектуру системи?

97.Які інструменти моніторингу ви використовували?

98.Що таке медіана та процентиль?

99.Що таке SLI, SLO, SLA? Навіщо це потрібно?

100.Архітектура системи для збору логів, ELK, EFK etc. Як зберегти логи у разі відмови сховища? Чи потрібно використовувати для цього брокер повідомлень? Чи потрібно робити throttling/rate limits?

101.Prometheus long-term storage. Які варіанти?

102.Як працює Prometheus?

103.У чому принципова відмінність між Grafana і Kibana?

104.В чому головна відмінність між Ansible and Terraform?

105.Що таке SAAS monitoring та які види знаєте?

106.Якщо ви використовуєте Datadog/NewRelic, то як нам відстежувати падіння інструментів моніторингу?

107.Що таке distributed tracing та error tracking systems? Як ви думаєте, коли варто їх використовувати?

Information Security

108.У чому різниця між RBAC та ABAC?

109.У чому полягає XSS атака? SQL injection? Що таке CSP?

110.Які базові заходи можна вжити для захисту SSH-з’єднання?

111.Root-пароль невідомий чи загублений. Яка процедура відновлення?

112.Як керувати правами на файловій системі в Linux?

113.Що таке Firewall?

114.Чим відрізняється stateless від stateful фаєрволів?

115.Скільки таблиць у iptables?

116.Чи можна налаштувати трансляцію NAT за допомогою iptables? Яку таблицю варто використати?

117.Яку таблицю використовують для зміни заголовків пакетів?

118.Якщо вам ламають Linux-сервер, то як більш ефективно блокувати трафік з IP-адрес?

119.Принцип роботи GCP Firewall: чи можемо ми профільтрувати трафік на Load Balancer?

120.Що таке SELinux?

121.Чи можна повністю від’єднати SELinux на льоту?

122.З якими secrets management systems ви працювали?

123.У нас є сервер NAT, і ми хочемо забезпечити доступ за ІР до сервера зовні. Як нам це реалізувати?

123.Щоб потрапити на сервер клієнта, треба залогінитись на 4+ jump хоста. Як це автоматизувати? Де ми зберігатимемо наш SSH-ключ?

Development

125.Що таке cookies? Для чого потрібні? JWT?

126.Що таке feature toggles та навіщо вони?

127.Що таке TDD (Test Driven Development) та BDD (Behaviour Driven Development)?

Databases

128.Що таке індекс і що таке ключ?

129.Які переваги та недоліки індексів?

130.Уявіть, що ви розробляєте систему білінгу, яка має обробляти тисячі рахунків. Яку стратегію оновлення даних ви б обрали?

131.Які методи найчастіше використовують для масштабування реляційних баз даних?

132.Опишіть механізм транзакцій БД.

133.Як ми можемо видалити таблицю чи базу даних?

134.Як знайти повільні запити у MySQL/PostgreSQL?

135.Які SQL-оператори маніпулювання даними ви знаєте?

136.Чи можна вивести список баз даних/таблиць через CLI? Як ми можемо перемикатися між базами даних MySQL/PostgreSQL?

137.Які storage engines в MySQL ви знаєте? Які відмінності?

138.Як реалізовано реплікацію MySQL master-master? Скільки серверів MySQL може бути залучено в такій взаємодії?

139.Як працює реплікація MySQL/PostgreSQL? Які параметри мають бути налаштовані для реплікації?

140.Порівняйте SQL і NoSQL.

141.Sharding vs replication?

142.Які є види індексів? Коли і для чого використовувати?

143.Вимоги до схеми БД. Character sets, collations, default, not null тощо.

144.Ми мігруємо MySQL/PostgreSQL з on-prem у хмару. Як нам зробити це з мінімальним даунтаймом?

145.Навіщо та як тестувати перформанс баз даних?

Практичні завдання

146.Напишіть Terraform module для інфраструктури тестового сервісу у AWS.

147.Напишіть hello-world програму мовою на ваш вибір і сформуйте для неї helm chart/kustomize.

148.Як організувати деплой без downtime?

149.Опишіть способи troubleshooting для Docker-контейнера.

150.Розібрати і пояснити структуру CI/CD pipeline (на прикладі gitlab.yml).

151.Продемонструйте навички роботи з GitOps, опишіть деплоймент простенької програми.

152.Як організувати деплой вебзастосунку, запущений на кількох серверах без (або з мінімальним) downtime?

153.Як за допомогою Ansible дізнатися default gateway для пулу серверів, і якщо він відрізняється від бажаного, записати рядок «hostname: gateway» у файл на локальній машині?