1. **SSH: co to je, co umožňuje, jak se konfiguruje, jaká je běžná konfigurace, na jakém běží portu.**

SSH (Secure Shell) je sieťový protokol, ktorý umožňuje bezpečnú komunikáciu medzi dvoma zariadeniami. Primárne sa používa na vzdialený prístup k serverom alebo zariadeniam, pričom zabezpečuje šifrovanie komunikácie, autentifikáciu a ochranu pred odpočúvaním alebo neoprávnenými zásahmi.

Čo umožňuje:

1. **Vzdialený prístup:** Bezpečný prístup k príkazovému riadku na vzdialenom serveri.
2. **Prenos súborov:** Pomocou nástrojov ako SCP alebo SFTP.
3. **Tunelovanie:** Presmerovanie portov na zabezpečenie prístupu k iným službám cez šifrovaný kanál.
4. **Autentifikácia:** Pomocou hesiel alebo kľúčov verejného a súkromného šifrovania.

Konfigurácia SSH:

1. **Konfiguračný súbor servera:** Nachádza sa obvykle na ceste /etc/ssh/sshd_config.
2. **Hlavné nastavenia:**
    - **Port:** Štandardne 22.
    - **PermitRootLogin:** Povolenie alebo zákaz prístupu pre root používateľa.
    - **PasswordAuthentication:** Povolenie autentifikácie heslom.
    - **PubkeyAuthentication:** Povolenie autentifikácie kľúčom.
    - **AllowUsers:** Zoznam povolených používateľov.
3. **Reštart služby:** Po zmene konfigurácie je potrebné reštartovať SSH službu:

sudo systemctl restart sshd

4. **Klientský konfiguračný súbor:** Zvyčajne sa nachádza v ~/.ssh/config a umožňuje preddefinovať nastavenia pre konkrétne servery:

Běžná konfigurácia:

- Štandardný port: 22.
- Povolený prístup pre konkrétnych používateľov (AllowUsers).
- Autentifikácia kľúčmi (zvýšená bezpečnosť).
- Zakázaný priamy prístup pre root používateľa (PermitRootLogin no).
- Logovanie spojení a pokusov o prihlásenie (nastavenie LogLevel).

Na akom porte beží:

- Predvolený port je **22**, ale môže byť zmenený pre zvýšenie bezpečnosti (napríklad na 2222).

Doplňujúca poznámka: Pri zmene portu v konfigurácii je potrebné prispôsobiť firewall

2. **SSH klíče: k čemu slouží, co umožňují, jak k nim přijdu, proč je chci používat a kde**

SSH kľúče sú nástroj na bezpečnú autentifikáciu pri používaní protokolu SSH. Fungujú na princípe asymetrického šifrovania a pozostávajú z privátneho kľúča (súkromného) a verejného kľúča.

K čomu slúžia:

1. **Autentifikácia:** Nahrádzajú tradičné prihlasovanie pomocou hesla.
2. **Zvýšenie bezpečnosti:** Heslá môžu byť uhádnuté alebo odpočúvané, kľúče sú oveľa bezpečnejšie.
3. **Automatizácia:** Umožňujú skripty a nástroje pristupovať k serverom bez interakcie používateľa (bez hesla).

Čo umožňujú:

1. **Bezheslový prístup:** Autentifikácia prebieha cez overenie kľúča bez nutnosti zadávať heslo.
2. **Autorizáciu konkrétnych kľúčov:** Môžete presne definovať, ktoré verejné kľúče majú prístup k serveru.
3. **Bezpečný prenos súborov:** Pomocou SCP alebo SFTP bez potreby hesla.
4. **Viacúrovňovú bezpečnosť:** Možno ich kombinovať s heslom (tzv. dvojfaktorová autentifikácia)

Jak k nim přijdu:

Generovanie SSH kľúčov: Používa sa príkaz ssh-keygen:

```
ssh-keygen -t rsa -b 4096 -C "tvoj_email@example.com"
```

Privátny kľúč: Uložený na vašom zariadení (napr. ~/.ssh/id_rsa).
Nikdy by sa nemal prenášať alebo zdieľať.
Môže byť chránený heslom pre dodatočné zabezpečenie.

Verejný kľúč: Ktorý sa nahráva na server (napr. ~/.ssh/id_rsa.pub).
Uložený na serveri v súbore ~/.ssh/authorized_keys.
Prenáša sa bezpečne cez ssh-copy-id alebo manuálne kopírovaním.

Prenos verejného kľúča na server:

```
ssh-copy-id user@server
```

Tento príkaz pridá váš verejný kľúč do súboru ~/.ssh/authorized_keys na serveri.

Proč je chci používat:

1. **Bezpečnosť:** Sú bezpečnejšie než heslá, pretože privátny kľúč je uložený len na vašom zariadení.
2. **Pohodlie:** Nemusíte si pamätať heslá pre každý server a zadávať ich pri každom prihlásení.
3. **Automatizácia:** SSH kľúče umožňujú jednoduchú správu serverov a procesov v skriptoch.
4. **Odolnosť voči útokom:** Útok hrubou silou je prakticky nemožný pri správnom použití.

SSH kľúče sú esenciálnou súčasťou bezpečného a efektívneho spravovania serverov, preto sa odporúčajú na všetkých kritických zariadeniach alebo systémoch.

3. **instalace software na Linuxovém stroji: jaké jsou možnosti, jaké jsou úskalí. (Tarball, balíky a repozitáře, AppImage, snapd, flatpak)**

Balíky a repozitáre (Debian/Ubuntu)

Balíky (napríklad .deb) obsahujú softvér s potrebnými súbormi a metadátami.
Softvér sa inštaluje pomocou **správcov balíkov**, ako je apt, ktorý sťahuje balíky z **oficiálnych repozitárov** alebo tretích zdrojov.

Príklad inštalácie:

- Aktualizácia repozitárov: sudo apt update
- Inštalácia balíka: sudo apt install názov_balíka

Výhody:

- Automatické riešenie závislostí (systém stiahne a nainštaluje všetky potrebné knižnice a softvér).
- Overené balíky, ktoré sú kompatibilné s konkrétnou distribúciou.
- Možnosť aktualizovať všetky nainštalované balíky naraz (sudo apt upgrade).

Nevýhody:

- Softvér v oficiálnych repozitároch nemusí byť vždy najnovšia verzia.
- Niektoré aplikácie môžu chýbať v štandardných repozitároch, a preto je potrebné pridať externé zdroje.

Tarball

Software distribuovaný ako archívy (.tar.gz, .tar.bz2) zvyčajne obsahuje zdrojový kód, ktorý je potrebné skompilovať.

Výhody:

- Dostupnosť najnovšej verzie softvéru priamo od vývojárov.
- Možnosť prispôsobiť proces kompilácie (napríklad špecifické funkcie).

Nevýhody:

- Manuálne riešenie závislostí, čo môže byť časovo náročné.
- Riziko inštalácie nesprávnych alebo nekompatibilných knižníc.
- Komplikovanejšie odinštalovanie.

Snap (Debian/Ubuntu)

Univerzálny balíčkovací systém vyvinutý spoločnosťou Canonical, ktorý umožňuje inštalovať softvér izolovaný od zvyšku systému (sandboxing).

Výhody:

- Aplikácie sú nezávislé od systémových knižníc, čo znižuje riziko konfliktov.
- Automatické aktualizácie.
- Funguje na väčšine distribúcií Linuxu.

Nevýhody:

- Vyššie nároky na úložisko a výkon, keďže každá aplikácia obsahuje všetky potrebné knižnice.

- Izolácia môže obmedziť prístup k niektorým systémovým zdrojom (napr. súborom mimo sandboxu).

4. **firewall: k čemu jsou, co jsou ty ??tables, jak se dají nakonfigurovat, ideální použití. Porty vs. Služby.**

Firewall je bezpečnostný nástroj na správu a kontrolu sieťovej komunikácie. Umožňuje obmedziť alebo povoliť prístup k určitým službám alebo portom na základe definovaných pravidiel.

Hlavné úlohy:

- Ochrana pred neoprávneným prístupom.
- Riadenie sieťovej prevádzky (povolenie/odmietnutie pripojení).
- Prevencia pred útokmi (napr. DDoS, skenovanie portov).
- Zabezpečenie interných sietí pred únikom dát.

Čo sú tie "tables" vo firewalle?

Linuxové firewally (napr. **iptables**, **nftables**) spracúvajú pravidlá pomocou **tabuliek**. Tabuľky určujú, aký typ akcií sa má vykonať s paketmi.

Základné tabuľky (iptables):

filter:

Štandardná tabuľka na povolenie alebo blokovanie prístupu.
Obsahuje reťazce:
INPUT: Rieši prichádzajúce pakety.
FORWARD: Rieši pakety, ktoré smerujú cez server (napr. router).
OUTPUT: Rieši odchádzajúce pakety.

nat (Network Address Translation):

Používa sa na úpravu cieľovej alebo zdrojovej adresy paketov.
Typické pri presmerovaní portov (DNAT) alebo zdieľaní internetu (SNAT).

mangle:

Na pokročilú manipuláciu s paketmi (napr. označovanie paketov pre smerovanie).

raw:

Používa sa na manipuláciu paketov predtým, ako prejdú cez sledovanie spojení.

Konfigurácia firewallu

Firewall sa konfiguruje pomocou nástrojov, ako sú **iptables**, **nftables**, alebo moderné nástroje vyššej úrovne, napr. **ufw** (Debian/Ubuntu) či **firewalld** (RedHat).

Príklady konfigurácie (iptables):

1. Zobrazenie aktuálnych pravidiel:
   sudo iptables -L -v
2. Povolenie pripojenia na port 22 (SSH):
   sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT
3. Zablokovanie všetkých prichádzajúcich spojení:
   sudo iptables -P INPUT DROP
4. Uloženie pravidiel:
   sudo iptables-save > /etc/iptables/rules.v4

Porty vs. Služby

- **Porty:**
    - Čísla (0–65535), ktoré označujú konkrétne aplikácie alebo služby na serveri.
    - Príklady: 22 (SSH), 80 (HTTP), 443 (HTTPS).
- **Služby:**
    - Aplikácie alebo procesy, ktoré počúvajú na konkrétnych portoch.
    - Napr. webový server Apache počúva na porte 80 (HTTP) a 443 (HTTPS).

Vzťah medzi portami a službami:

- Firewall môže obmedzovať komunikáciu buď podľa portu (napr. blokovať port 80), alebo podľa služby (napr. blokovať Apache).
- **Výhody blokovania portov:** Nezáleží na tom, aká služba je aktívna na danom porte.
- **Výhody blokovania služieb:** Umožňuje presné cielenie, ale vyžaduje znalosť procesov.

Ideálne použitie firewallu

1. **Bezpečnostné zásady:**
    - Povoliť len tie porty/služby, ktoré sú nevyhnutné.
    - Zaviesť pravidlo "deny by default" (blokovať všetko, čo nie je povolené).
2. **Praktické scenáre:**
    - Servery: Povoliť len nevyhnutné porty (napr. 22 pre SSH, 443 pre HTTPS).
    - Desktopové použitie: Použiť nástroje ako **ufw**, ktoré umožňujú jednoduchú konfiguráciu.
    - Router: Použiť NAT a filtrovať prichádzajúce pakety.

5. **nastavení sítě: IPv4 a IPv6. Jak nastavím jedno nebo druhé na serveru. Jak musím řešit firewall a pro koho s ohledem na IPv4 a IPv6. Lze nějakou vynechat?**

IPv4 používa 32-bitové adresy, ktoré sú zapísané v štyroch číslach oddelených bodkami (napr. 192.168.1.1). Je najrozšírenejší, ale počet adries je obmedzený a často sa používa NAT na ich úsporu.
IPv6 používa 128-bitové adresy zapisované hexadecimálne a oddelené dvojbodkami (napr. 2001:db8::1). Poskytuje obrovský priestor adries a umožňuje priamu komunikáciu medzi zariadeniami bez potreby NAT.

Ako nastaviť IPv4 alebo IPv6 na serveri

1. **IPv4:**
    - Bežne sa nastavuje statická adresa, maska podsiete, brána a DNS server.
    - Používa sa pre pripojenie v menších sieťach alebo tam, kde IPv6 nie je podporované.
2. **IPv6:**
    - Nastavenie zahŕňa pridelenie adresy, prefix (zvyčajne 64 bitov), bránu a DNS servery.
    - Pri väčších sieťach alebo moderných aplikáciách je často potrebné zabezpečiť podporu IPv6.

Obe verzie môžu byť nastavené manuálne (staticky) alebo dynamicky cez DHCP.

Firewall pre IPv4 a IPv6

- Firewall musí byť nakonfigurovaný samostatne pre IPv4 aj IPv6, pretože ide o dva rôzne protokoly.
- Pri IPv4 sa pravidlá týkajú portov a adresného priestoru 32 bitov. Pri IPv6 je potrebné riešiť širší priestor a ďalšie vlastnosti (napr. multicast).
- Ak IPv6 nie je správne nakonfigurovaný alebo filtrovaný, môže byť obchádzaný firewall určený len pre IPv4.

Porty vs. Služby

- **Porty:** Sú to čísla, ktoré identifikujú konkrétne aplikácie alebo služby na serveri. Napríklad port 22 sa používa pre SSH a port 80 pre HTTP.
- **Služby:** Predstavujú samotné aplikácie (napr. webový server alebo SSH server), ktoré využívajú porty na komunikáciu.

Firewall môže obmedzovať prístup buď podľa portov, alebo podľa služieb. Napríklad môžete zablokovať všetku komunikáciu na porte 80 (HTTP), čím efektívne znemožníte prístup k webovému serveru.

Je možné niektorú vynechať?

- **IPv6:** Ak nie je potrebné, môžete IPv6 vypnúť. Niektoré staršie siete alebo zariadenia ho nepodporujú. V moderných prostrediach sa však odporúča, aby bol povolený.
- **IPv4:** Prakticky ho nemožno vypnúť, pretože väčšina služieb a sietí stále funguje primárne na ňom.

Odporúčania

- Používajte obe verzie, ak je to možné, aby ste boli pripravení na moderné siete.
- Pri nastavovaní firewallu vždy zaistite, že pravidlá pokrývajú aj IPv4, aj IPv6, aby sa zabránilo potenciálnym bezpečnostným rizikám.

6. **logovací soubory: kde je najdu a co v nich najdu. Příklad aspoň 4 souborů a proč si všímám zrovna jich. Jak probíhá rotace logů a co je vzdálené logování.**

Logy (záznamy) sú textové súbory, ktoré obsahujú dôležité informácie o udalostiach v systéme. Slúžia na monitorovanie chodu systému, hľadanie chýb, analýzu bezpečnostných incidentov a auditovanie.

Kde logy nájdem

V Linuxových distribúciách sa väčšina logov nachádza v adresári /var/log. Medzi základné súbory patria:

1. /var/log/syslog alebo /var/log/messages (závisí od distribúcie):
    - Obsahuje všeobecné systémové udalosti.
    - Monitoruje aktivitu systému, služieb a jadra.
    - Príklad použitia: Hľadanie chýb pri štarte systému.
2. /var/log/auth.log:
    - Zaznamenáva udalosti týkajúce sa autentifikácie (prihlásenie/odhlásenie používateľov, SSH pokusy).
    - Príklad použitia: Kontrola podozrivých prihlásení alebo neúspešných pokusov.
3. /var/log/kern.log:
    - Obsahuje správy generované jadrom systému.
    - Príklad použitia: Diagnostika hardvérových problémov (napr. chyby ovládačov).
4. /var/log/dpkg.log (Debian/Ubuntu):
    - Uchováva informácie o inštaláciách, aktualizáciách a odstraňovaní balíkov cez správcu balíkov apt.
    - Príklad použitia: Kontrola, aký softvér bol nedávno nainštalovaný alebo odstránený.

Prečo si všímať tieto logy

1. syslog: Základný log pre sledovanie všeobecného stavu systému a služieb.
2. auth.log: Nevyhnutný pre bezpečnostné kontroly a odhalenie neoprávnených prístupov.
3. kern.log: Dôležitý pre riešenie hardvérových alebo systémových problémov.
4. dpkg.log: Pomáha pri správe softvéru a sledovaní zmien v systéme.

Každý z týchto logov pokrýva inú oblasť, čo umožňuje efektívnu diagnostiku rôznych problémov.

Ako prebieha rotácia logov

Logovacie súbory môžu rásť na veľkosti a zaplniť disk. Na ich správu sa používa **rotácia logov**, ktorá:

1. **Rozdeľuje logy na staré a nové**:
    - Staré logy sa premenúvajú (napr. syslog.1, syslog.2.gz).
    - Počet uchovávaných verzií je obmedzený.
2. **Komprimuje staré logy**:
    - Znižuje sa tak miesto, ktoré zaberajú (napr. gzip).
3. **Maže veľmi staré logy**:
    - Keď prekročia definovaný počet verzií.

Konfigurácia rotácie logov je v súbore /etc/logrotate.conf a prípadne v jednotlivých konfiguráciách aplikácií.

Logovací soubory v Linuxu

Logy (záznamy) sú textové súbory, ktoré obsahujú dôležité informácie o udalostiach v systéme. Slúžia na monitorovanie chodu systému, hľadanie chýb, analýzu bezpečnostných incidentov a auditovanie.

Vzdialené logovanie

Vzdialené logovanie umožňuje posielať logy zo servera na iný server. Je užitočné na:

- **Centralizované ukladanie logov**: Záznamy z viacerých zariadení sa ukladajú na jednom mieste.
- **Bezpečnosť**: Ak dôjde k narušeniu systému, lokálne logy môžu byť upravené, ale vzdialené logy zostanú nedotknuté.
- **Analýzu a škálovanie**: Vzdialené logy sa môžu spracovávať pomocou nástrojov (napr. Elasticsearch, Splunk).

Protokolom používaným na vzdialené logovanie je najčastejšie **syslog** alebo **rsyslog**. Logy sa posielajú cez sieť na definovaný vzdialený server.

7. **ftp: na co je, komu slouží, jak se instaluje, konfiguruje, jaké je běžné nastavení**

FTP (File Transfer Protocol) je protokol určený na prenos súborov medzi klientom a serverom cez sieť. Umožňuje nahrávanie, sťahovanie, mazanie a správu súborov na vzdialenom serveri.

- **Na co je:**
    - Prenos súborov medzi zariadeniami.
    - Zdieľanie dát v rámci firmy alebo verejne dostupné súbory.
    - Administrácia súborov na vzdialenom serveri.
- **Komu slouží:**
    - Správcovia serverov: Na správu súborov na vzdialenom zariadení.
    - Vývojári: Na nahrávanie a správu webových stránok.
    - Bežní používatelia: Na sťahovanie súborov z verejných FTP serverov.

Jak se FTP instaluje a konfiguruje

Instalace FTP serveru (Debian/Ubuntu)

1. **Výběr softwaru:**
    - Najpoužívanejšie FTP servery sú:
        - **vsftpd**: Jednoduchý a bezpečný.
        - **ProFTPD**: Flexibilný a dobre rozšíriteľný.
2. **Instalace vsftpd:**

    sudo apt update
    sudo apt install vsftpd

Základní konfigurace
Konfigurácia FTP servera sa nastavuje v súbore /etc/vsftpd.conf.

- Povolenie anonymného prístupu - ak chceme povoliť prístup bez prihlásenia (verejné FTP)
- Prístup len pre registrovaných užívateľov - povoliť len prihlásenie používateľov so systémovým účtom
- Zakázanie zápisu pre anonymných používateľov - ak nechceme, aby anonymní používatelia mohli nahrávať súbory
- Nastavenie domovského adresára - Povolenie používateľom vidieť iba svoj adresár

Běžné nastavení

1. **Verejný FTP server:**
    - Povolený anonymný prístup pre čítanie súborov.
    - Zákaz zápisu pre anonymných používateľov.
2. **Súkromný FTP server:**
    - Prístup len pre registrovaných používateľov.
    - Zabezpečený prenos dát pomocou TLS.
    - Obmedzenie prístupu do určených adresárov (chroot).
3. **Porty:**
    - FTP beží štandardne na porte **21**.
    - Pre pasívny režim je potrebné povoliť rozsah dynamických portov (napr. 40000–50000)

Bezpečnostní úvahy

1. **Šifrovanie:**
    - FTP posiela heslá a dáta v čistej podobe. Odporúča sa používať **FTPS** (FTP cez TLS) alebo **SFTP** (Secure FTP cez SSH).
2. **Ochrana proti brute-force útokom:**
    - Použitie nástrojov ako **fail2ban**, ktoré blokujú neúspešné pokusy o prihlásenie.
3. **Firewall:**
    - Povoliť iba porty 21 a rozsah pre pasívny režim.
    - Nastaviť pravidlá pre povolené IP adresy, ak je to potrebné.

Alternatívy k FTP

- **SFTP:** Prenos súborov cez SSH, bezpečnejší a jednoduchší na konfiguráciu.
- **SCP:** Jednoduché a rýchle kopírovanie súborov cez SSH.
- **HTTP/HTTPS:** Na distribúciu verejných súborov je často efektívnejšie použiť webový server.

FTP je tradičný a stále populárny protokol, ale v moderných aplikáciách sa často nahrádza bezpečnejšími alternatívami, najmä ak je dôležitá ochrana dát.

8. **samba: na co je, komu slouží, jak se instaluje, konfiguruje, jaké je běžné nastavení**

Samba je softvérový balík pre Linux/Unix, ktorý umožňuje zdieľanie súborov a tlačiarní medzi systémami Linux/Unix a Windows. Využíva protokol SMB/CIFS (Server Message Block/Common Internet File System), ktorý je natívny pre Windows.

Na co je:

- Umožňuje Linuxovým serverom fungovať ako **súborové servery** pre Windows.
- Zdieľanie tlačiarní medzi Linuxom a Windows.
- Integrácia Linuxu do **Windows domén** (Active Directory).

Komu slouží:

1. **Firmám:**
    - Na centralizované zdieľanie súborov medzi Windows a Linux zariadeniami.
2. **Domácim používateľom:**
    - Na zdieľanie dát medzi domácimi počítačmi s rôznymi operačnými systémami.
3. **Správcom serverov:**
    - Na zabezpečenie prístupu k súborom a tlačiarňam vo zmiešanom prostredí

Jak se instaluje Samba

```
sudo apt install samba
```

Jak se Samba konfiguruje

Konfigurácia Samby sa vykonáva úpravou súboru /etc/samba/smb.conf.

1. **Základní struktura konfigurace:**
    - **Globální sekce:** Nastavenia, ktoré ovplyvňujú celý server.
    - **Sdílené sekce:** Definujú konkrétne zdieľané adresáre a ich prístupové práva.

Bezpečnostní úvahy

- **Použitie hesiel:** Ak je zdieľanie citlivé, vždy povolte autentifikáciu používateľov.
- **Firewall:** Povoliť porty 137-139 (NetBIOS) a 445 (SMB).
- **Šifrovanie:** Povoliť šifrovaný prenos dát, ak je zdieľanie dostupné cez verejnú sieť.

Shrnutí

Samba je kľúčový nástroj na zdieľanie súborov a tlačiarní v zmiešanom prostredí Linux/Windows. Jej konfigurácia môže byť jednoduchá pre základné zdieľanie, ale umožňuje aj pokročilé nastavenia pre komplexné firemné siete.

9. **NFS share: na co je, komu slouží, jak se instaluje, konfiguruje, jaké je běžné nastavení**

NFS (Network File System) je protokol umožňujúci zdieľanie súborov medzi systémami Linux/Unix prostredníctvom siete. NFS umožňuje, aby klientský systém pristupoval k súborom na vzdialenom serveri, ako keby boli lokálne.

Na co je:

- **Zdieľanie súborov**: Používa sa na sprístupnenie adresárov vzdialeným zariadeniam.
- **Centralizované úložisko**: Poskytuje centrálnu správu súborov, napríklad v organizáciách.
- **Práca na rovnakých dátach**: Umožňuje viacerým zariadeniam spolupracovať na rovnakých súboroch.

Komu slouží:

1. **Firmám:** Na centrálne úložisko, ku ktorému majú prístup všetci používatelia.
2. **Serverovým administrátorom:** Na zdieľanie dát medzi servermi v rovnakej sieti.
3. **Vývojárom:** Na prácu s rovnakými dátami na viacerých zariadeniach.

Jak se instaluje NFS

```
sudo apt install nfs-kernel-server
```

Jak se konfiguruje NFS

- **Na serveru:**
  - Konfigurácia NFS sa vykonáva v súbore /etc/exports
  - **/srv/nfs/shared**: Adresár, ktorý chcete zdieľať.
  - 192.168.1.0/24: Rozsah IP adries, ktorým je prístup povolený.
  - **Možnosti:**
    - rw: Povolenie čítania aj zápisu.
    - sync: Dáta sa zapisujú okamžite (vyššia bezpečnosť).
    - no_subtree_check: Znižuje riziko problémov so súbormi mimo zdieľaného adresára.
- **Na klientovi:**
  - Vytvorte pripojovací bod (napr. /mnt/nfs)

Běžné nastavení NFS

1. Zdieľanie len na čítanie (read-only) - vhodné pre verejné dáta, kde nechcete povoliť úpravy
2. Zdieľanie s plnými právami - pre interné použitie, kde majú používatelia práva na čítanie aj zápis
3. Prístup obmedzený na konkrétnu IP adresu - ak chcete povoliť prístup len pre konkrétne zariadenie

Bezpečnostní úvahy

1. **Obmedzenie prístupu:** Povoliť prístup iba z dôveryhodných IP adries.
2. **Firewall:** Povoliť len potrebné porty (2049 pre NFS).
3. **Šifrovanie:** NFS v základnej konfigurácii nešifruje prenos. Pre citlivé dáta je vhodné kombinovať ho s VPN alebo šifrovaním na úrovni súborového systému.

Výhody NFS

- Transparentné zdieľanie súborov medzi zariadeniami.
- Jednoduché použitie v homogénnom prostredí Linux/Unix.
- Možnosť automatického pripojenia po štarte.

Shrnutí

NFS je ideálny na zdieľanie súborov v lokálnej sieti. Je jednoduchý na konfiguráciu a efektívny na prenos dát medzi servermi a klientmi. Běžná konfigurácia zahŕňa prístup len na čítanie alebo plné práva s autentifikovaným prístupom. V modernom prostredí je dôležité dbať na bezpečnosť a obmedziť prístup na dôveryhodné zariadenia.

10. **web server apache: co to je, proč je to dobré, jak ho použít, co od něj čekat**

Apache HTTP Server (často jen Apache) je jeden z nejrozšířenějších open-source webových serverů na světě. Umožňuje hostování webových stránek a aplikací. Je robustní, flexibilní a podporuje širokou škálu funkcionalit prostřednictvím modulů.

Proč je Apache dobrý

1. **Flexibilita:**
    - Podpora různých technologií (např. PHP, Python, Perl).
    - Možnost rozšíření pomocí modulů, jako je **mod_rewrite** (přepis URL), **mod_ssl** (HTTPS), nebo **mod_proxy** (reverzní proxy).
2. **Široká podpora:**
    - Funguje na většině operačních systémů (Linux, Windows, macOS).
    - Komunita a dokumentace jsou rozsáhlé, což usnadňuje řešení problémů.
3. **Stabilita a výkon:**
    - Je ověřený dlouholetým používáním v produkčních prostředích.
    - Nabízí optimalizace pro statický i dynamický obsah.
4. **Bezpečnost:**
    - Podpora šifrovaného přenosu (HTTPS pomocí TLS).
    - Konfigurační možnosti pro řízení přístupu a ochranu dat.
5. **Modularita:**
    - Možnost načíst jen ty moduly, které jsou potřebné, což zvyšuje efektivitu.

Jak ho použít

1. **Instalace:**
   sudo apt install apache2

2. **Základní spuštění:**
   Po instalaci je Apache standardně spuštěný a dostupný na portu **80** (HTTP).
   Webový obsah se obvykle ukládá do adresáře /var/www/html.

3. **Konfigurace:**
   Hlavní konfigurační soubor:
   Debian/Ubuntu: /etc/apache2/apache2.conf

   Virtuální hostitelé (pro hostování více webů):
   /etc/apache2/sites-available na Debian/Ubuntu

Co od něj čekat

1. **Funkčnost:**
    - Apache zajišťuje dostupnost webových stránek a aplikací 24/7.
    - Podpora statických (HTML, CSS) i dynamických aplikací (PHP, Python).
2. **Konfigurace přístupu:**
    - Umožňuje řídit přístup podle IP adres, uživatelů nebo přes certifikáty (HTTPS).
3. **Škálovatelnost:**

- Dobře funguje v malých i velkých produkčních prostředích.
- Lze ho kombinovat s dalšími nástroji, jako je **reverzní proxy** (např. Nginx).

4. **Bezpečnost:**
    - Možnost šifrování přenosu (HTTPS pomocí Let's Encrypt).
    - Logování přístupů a chyb, což usnadňuje audit a řešení problémů.

Kdy ho používat

- **Malé projekty:** Jednoduché weby nebo intranetové aplikace.
- **Střední a velké systémy:** Hostování komplexních aplikací s podporou různých technologií.
- **Hybridní infrastruktura:** Apache lze kombinovat s Nginx jako reverzní proxy pro zvýšení výkonu.

Apache je výkonný a všestranný webový server, který je ideální pro hostování všeho od malých osobních webů až po velké firemní aplikace. Díky své modulární architektuře a široké podpoře je vhodnou volbou v mnoha různých scénářích.

11. **web server nginx: : co to je, proč je to dobré, jak ho použít, co od něj čekat**

Nginx (vyslovuje se jako "Engine-X") je moderní open-source webový server a reverzní proxy server, navržený pro vysoký výkon a nízkou spotřebu paměti. Nginx je známý svou schopností zvládat velké množství současných připojení a je často využíván v prostředích s vysokou zátěží.

Proč je Nginx dobrý

1. **Vysoký výkon:**
    - Asynchronní, event-driven architektura umožňuje efektivní obsluhu tisíců připojení najednou.
    - Lepší výkon při zpracování statického obsahu oproti tradičnímu Apache.
2. **Nízká spotřeba zdrojů:**
    - Vhodný pro servery s omezenými hardwarovými prostředky.
    - Optimalizovaný pro nízkou paměťovou stopu.
3. **Flexibilní použití:**
    - Může fungovat jako webový server, reverzní proxy nebo load balancer.
4. **Bezpečnost:**
    - Podpora šifrování (TLS/SSL) s moderními protokoly a bezpečnostními standardy.
    - Možnost omezení přístupu a řízení šířky pásma.
5. **Škálovatelnost:**
    - Snadná konfigurace pro horizontální i vertikální škálování aplikací.

Co od Nginx čekat

1. **Vynikající výkon:**
    - Ideální pro zpracování statického obsahu (HTML, CSS, JavaScript).
    - Vynikající efektivita při práci jako reverzní proxy nebo load balancer.
2. **Podpora moderních technologií:**
    - HTTP/2, TLS 1.3, Gzip komprese, a další moderní standardy.
3. **Škálovatelnost:**
    - Vhodný pro prostředí s vysokou zátěží nebo velkým počtem požadavků.
    - Snadná integrace s cloudovými službami a kontejnery (Docker).
4. **Bezpečnostní funkce:**
    - Šifrování dat přes HTTPS.
    - Možnost omezení přístupu podle IP adresy nebo jiných pravidel.
5. **Flexibilita v nasazení:**
    - Možnost použití jako samostatný webový server nebo jako reverzní proxy před jinými servery (např. Apache, aplikace v Node.js nebo PHP).

Kdy ho používat

- **Pro statické weby:** Nejlepší volba díky vysoké efektivitě.
- **Reverzní proxy:** Ideální pro směrování požadavků na backend servery nebo load balancing.
- **Moderní aplikace:** Nginx je dobře přizpůsoben pro cloudové prostředí a moderní standardy.

Shrnutí: Nginx je výkonný a efektivní webový server, který se nejlépe hodí pro prostředí s vysokým výkonem. Díky své flexibilitě, nízké spotřebě zdrojů a podpoře moderních technologií je

preferovanou volbou pro mnoho webových aplikací, od malých projektů až po velké produkční nasazení.

12. **co je to LAMP server a proč ty písmenka LAMP už nejsou aktuální**

LAMP je akronym pro stack technologií používaných k nasazení dynamických webových aplikací. Skládá se z následujících komponent:

1. **L - Linux**: Operační systém, na kterém běží celý stack.
2. **A - Apache**: Webový server, který obsluhuje požadavky na webové stránky.
3. **M - MySQL**: Databázový systém používaný pro ukládání dat aplikace.
4. **P - PHP**: Programovací jazyk pro dynamický obsah (alternativně Perl nebo Python).

Proč je LAMP server důležitý

- **Jednoduchost:** Snadná konfigurace a integrace jednotlivých komponent.
- **Open-source:** Všechny technologie v LAMP stacku jsou zdarma a mají širokou komunitní podporu.
- **Flexibilita:** LAMP server umožňuje hostovat různé typy dynamických webových aplikací, například systémy pro správu obsahu (CMS) jako WordPress, Joomla nebo Drupal.
- **Stabilita:** Linux a Apache jsou ověřené technologie s dlouholetým nasazením v produkci.

Proč písmenka LAMP už nejsou aktuální

Technologický svět se mění a klasický LAMP stack není vždy nejvhodnějším řešením. Jednotlivé komponenty jsou stále častěji nahrazovány alternativami:

1. **Linux**:
    - Linux je stále dominantní, ale objevují se další alternativy, jako jsou kontejnery (Docker) nebo serverless platformy.
2. **Apache**:
    - Apache zůstává populární, ale v mnoha případech je nahrazován **Nginx** díky jeho lepšímu výkonu a nižší spotřebě zdrojů.
    - V moderních aplikacích se někdy používají integrované webové servery v rámci programovacích jazyků (např. Node.js).
3. **MySQL**:
    - MySQL je často nahrazováno alternativami, jako je:
        - **MariaDB** (fork MySQL s lepšími licenčními podmínkami a rozšířeními).
        - **PostgreSQL** (pokročilejší funkce a větší flexibilita).
        - **NoSQL databáze**, jako je MongoDB nebo CouchDB, pro moderní aplikace pracující s nestrukturovanými daty.
4. **PHP**:
    - PHP je stále běžné, ale mnoho moderních aplikací a vývojářů přechází na jiné technologie:
        - **Node.js** pro asynchronní aplikace.
        - **Python** s frameworky jako Django nebo Flask.
        - **Ruby** s frameworkem Ruby on Rails.

Moderní variace LAMP stacku

1. **LEMP**:

    - Nahrazení Apache za Nginx (Linux + Nginx + MySQL/MariaDB + PHP/Python/Perl).
2. **MEAN**:
    - Kompletně moderní stack: MongoDB + Express.js + Angular.js + Node.js.
3. **JAM**:
    - Pro statické weby: JavaScript + API + Markup.
4. **Serverless stacky**:
    - Použití cloudových služeb místo tradičních serverových technologií.

LAMP server byl dlouho standardem pro nasazení webových aplikací díky své jednoduchosti, stabilitě a open-source povaze. Dnes už však není vždy nejlepší volbou, protože moderní aplikace často vyžadují vyšší výkon, flexibilitu nebo používají nové přístupy, jako jsou NoSQL databáze, kontejnery nebo serverless architektura. Písmena LAMP jsou tedy stále relevantní, ale často nahrazována alternativními technologiemi.

13. **skriptovací jazyky pro webserver: jaké znáte, co dělají, proč je chci, jak je připojím k webserveru**

Skriptovací jazyky umožňují generovat dynamický obsah na webových stránkách a zpracovávat data na serverové straně. Webové servery, jako jsou Apache nebo Nginx, často podporují několik skriptovacích jazyků, aby splnily různé potřeby vývojářů.

Jaké skriptovací jazyky známe

1. **PHP**:
    - Nejoblíbenější skriptovací jazyk pro weby, jako je WordPress, Joomla nebo Drupal.
    - Výhody: Jednoduchá syntax, široká podpora a obrovská komunita.
    - Použití: Dynamické generování HTML, zpracování formulářů, práce s databázemi (MySQL/MariaDB).
2. **Python**:
    - Flexibilní jazyk používaný s frameworky jako Django nebo Flask.
    - Výhody: Čistá syntax, robustní knihovny, vhodný pro složitější aplikace.
    - Použití: API, webové aplikace, analýza dat.
3. **JavaScript (Node.js)**:
    - JavaScript na straně serveru pomocí Node.js.
    - Výhody: Asynchronní zpracování, ideální pro aplikace s vysokou zátěží (chaty, streaming).
    - Použití: Real-time aplikace, REST API, microservices.
4. **Ruby**:
    - Používá framework Ruby on Rails.
    - Výhody: Snadno čitelná syntax, rychlý vývoj.
    - Použití: Webové aplikace, e-shopy, projekty s důrazem na rychlé prototypování.
5. **Perl**:
    - Tradiční jazyk s širokým využitím v síťových aplikacích.
    - Výhody: Velká síla v manipulaci textu, vhodný pro specifické úlohy.
    - Použití: Administrace, reporty, práce se soubory.
6. **Java**:
    - Používá se na serverech s Java Servlet Container (např. Tomcat).
    - Výhody: Silná typová kontrola, škálovatelnost.
    - Použití: Podnikové aplikace, složité systémy.

Co skriptovací jazyky dělají

- **Generují dynamický obsah**: Vytvářejí webové stránky na základě uživatelských požadavků nebo dat z databází.
- **Zpracovávají formuláře**: Přijímají a validují data od uživatelů.
- **Pracují s databázemi**: Připojují se k databázím a provádějí dotazy (např. SELECT, INSERT).
- **API a integrace**: Umožňují komunikaci mezi různými aplikacemi.
- **Bezpečnost**: Zajišťují autentifikaci a autorizaci uživatelů.

Proč je chci používat

1. **Dynamický obsah**:
    - Statické stránky nestačí pro moderní aplikace.
    - Skriptovací jazyky umožňují personalizaci (např. zobrazení přihlášeného uživatele).
2. **Integrace s databázemi**:
    - Ukládání a zpracování dat je klíčové pro většinu aplikací.
3. **Rozšíření funkcionality**:
    - Snadno lze přidat interaktivní prvky, jako je zpracování formulářů, odesílání e-mailů nebo generování reportů.
4. **Flexibilita a škálovatelnost**:
    - Skriptovací jazyky podporují různé potřeby, od malých osobních webů až po složité podnikové systémy.

Skriptovací jazyky jsou nepostradatelné pro moderní webové aplikace. Volba konkrétního jazyka závisí na požadavcích projektu. Webové servery jako Apache a Nginx podporují různé jazyky buď přímo (Apache), nebo přes aplikační servery (Nginx). Skriptovací jazyky umožňují vytvářet dynamický obsah, integrovat se s databázemi a rozšiřovat funkcionalitu aplikací.

14. **HTTPS: proč a jak jej zařídíte, co je CA. Jak vypadá výměna klíčů, jak probíhá instalace certifikátu**

HTTPS (Hypertext Transfer Protocol Secure) je šifrovaná verze protokolu HTTP, která zajišťuje bezpečný přenos dat mezi klientem (prohlížečem) a serverem.

1. **Šifrování:**
    - Zabraňuje odposlechu dat třetí stranou.
    - Citlivá data, jako jsou hesla nebo osobní údaje, jsou chráněna.
2. **Autentizace:**
    - Pomocí certifikátů se ověřuje, že se uživatel připojuje ke správnému serveru.
3. **Integrita:**
    - Zajišťuje, že data nebyla během přenosu změněna nebo narušena.
4. **SEO výhoda:**
    - Vyhledávače (např. Google) preferují HTTPS stránky ve výsledcích vyhledávání.

Jak HTTPS zařídit

1. **Certifikát SSL/TLS:**
    - HTTPS vyžaduje instalaci certifikátu na webový server.
    - Certifikát je vydán certifikační autoritou (**CA**, viz níže).
2. **Podpora serveru:**
    - Webový server (např. Apache, Nginx) musí mít povolenou podporu TLS/SSL.

Co je CA (Certifikační autorita)

Certifikační autorita (**Certificate Authority**) je důvěryhodná organizace, která vydává digitální certifikáty.

- **Úloha CA:**
    - Ověřuje identitu majitele domény nebo organizace.
    - Vydává certifikát, který potvrzuje pravost serveru.
    - Certifikát je důvěryhodný díky podpisu CA.
- **Známé CA:**
    - Let’s Encrypt (zdarma, automatizované vydání certifikátů).
    - DigiCert, GlobalSign, Sectigo (komerční certifikáty).

Jak vypadá výměna klíčů

HTTPS využívá asymetrickou kryptografii pro výměnu klíčů a symetrickou kryptografii pro šifrování dat.

1. **Navázání spojení (TLS Handshake):**
    - Klient (prohlížeč) odešle požadavek na server a uvede podporované šifrovací algoritmy.
    - Server odpoví s veřejným klíčem a certifikátem.
    - Klient ověří certifikát pomocí kořenových certifikátů CA.
    - Klient a server se dohodnou na společném šifrovacím klíči.
2. **Výměna klíčů:**
    - Klient zašifruje svůj klíč (session key) veřejným klíčem serveru.

- Server tento klíč dešifruje pomocí svého privátního klíče.
- Klient i server následně komunikují pomocí tohoto symetrického klíče.

3. **Začátek šifrované komunikace:**
    - Data jsou šifrována a dešifrována pomocí sjednaného symetrického klíče.

Jak probíhá instalace certifikátu

1. **Získání certifikátu:**
    - Zaregistrujte doménu u certifikační autority (CA).
    - Vygenerujte **CSR (Certificate Signing Request)**:
        - Obsahuje informace o doméně a veřejný klíč.
    - Certifikát obdržíte od CA po ověření vaší identity.
2. **Instalace na server:**
    - U Apache:
        - Certifikát a privátní klíč umístíte do /etc/ssl/certs/ a /etc/ssl/private/.
        - Upravíte konfigurační soubor
        - Restart serveru

3. **Ověření certifikátu:**
    - Zkontrolujte, zda je HTTPS správně nastaveno

HTTPS je nezbytný pro bezpečný přenos dat, autentizaci serveru a ochranu před útoky. K dosažení HTTPS potřebujete certifikát SSL/TLS vydaný certifikační autoritou (CA). Instalace certifikátu na server zahrnuje správu klíčů, konfiguraci serveru a ověření certifikátu. Díky nástrojům jako Let's Encrypt je dnes zavedení HTTPS jednoduché a široce dostupné.

15. **mailserver: co dělá mailserver, jak ho nainstaluju, co pomocí něj můžu provádět. Jaké jsou protokoly pro přijímání a posílání mailů**

Mailserver je software, který umožňuje odesílání, přijímání a ukládání e-mailových zpráv. Je klíčovou součástí infrastruktury elektronické pošty a zajišťuje komunikaci mezi uživateli přes síť.

- **Primární funkce:**
    1. **Příjem e-mailů** od jiných mailserverů a jejich doručení do schránek uživatelů.
    2. **Odesílání e-mailů** od uživatelů na jiné mailservery.
    3. **Ukládání e-mailů**, aby si je uživatelé mohli stáhnout nebo přečíst.
- **Hlavní komponenty mailserveru:**
    1. **MTA (Mail Transfer Agent):** Přenáší e-maily mezi servery (např. Postfix, Sendmail, Exim).
    2. **MDA (Mail Delivery Agent):** Doručuje e-maily do schránek uživatelů (např. Dovecot, Procmail).
    3. **MUA (Mail User Agent):** Klientské aplikace, které komunikují s mailserverem (např. Thunderbird, Outlook).

jak ho nainstaluju

1. **Výběr softwaru:**
    . MTA: Postfix, Sendmail, Exim.
    . MDA: Dovecot, Courier.
    . Webmail: Roundcube, RainLoop (volitelné pro přístup přes web).
2. **Instalace na Debian/Ubuntu:**
    . sudo apt update
    . sudo apt install postfix dovecot-core dovecot-imapd
3. **Základní konfigurace Postfixu:**
    . Při instalaci budete vyzváni k výběru typu konfigurace:
    . Internet Site: Mailserver přijímá a odesílá poštu přímo přes internet.
    . Nastavte název systému (např. mail.example.com).
4. **Základní konfigurace Dovecotu (pro IMAP):**

Co pomocí něj můžu provádět

1. **Odesílání e-mailů:**
    - Přes SMTP můžete odesílat zprávy ostatním uživatelům.
    - Použití MUA (např. Thunderbird) nebo příkazového řádku.
2. **Přijímání e-mailů:**
    - Mailserver přijímá poštu přes SMTP a ukládá ji do schránek uživatelů.
3. **Správa schránek:**
    - Přístup ke zprávám pomocí protokolů IMAP nebo POP3.
    - Možnost třídění, filtrování a ukládání zpráv.
4. **Zabezpečení pošty:**
    - Podpora šifrovaného přenosu (TLS pro SMTP, IMAP, POP3).
    - Použití antispamových a antivirových nástrojů (např. SpamAssassin, ClamAV).

Jaké jsou protokoly pro přijímání a posílání mailů

1. **Pro odesílání e-mailů:**
    - **SMTP (Simple Mail Transfer Protocol):**
        - Zajišťuje odesílání e-mailů z klienta na server a mezi servery.
        - Standardně běží na portu **25**, pro zabezpečené připojení se používá port **587** nebo **465** (s TLS).
2. **Pro přijímání e-mailů:**
    - **POP3 (Post Office Protocol, verze 3):**
        - Stahuje e-maily ze serveru na klienta a obvykle je ze serveru maže.
        - Jednoduchý protokol, běží na portu **110** (nezabezpečené) nebo **995** (s TLS).
        - Vhodný pro případy, kdy není potřeba mít e-maily uložené na serveru.
    - **IMAP (Internet Message Access Protocol):**
        - Umožňuje číst e-maily přímo na serveru bez jejich mazání.
        - Vhodný pro přístup z více zařízení.
        - Běží na portu **143** (nezabezpečené) nebo **993** (s TLS).

Proč chci mít mailserver

1. **Kontrola nad e-maily:**
    - Umožňuje spravovat poštu bez závislosti na externích poskytovatelích.
    - Vhodné pro firmy a organizace, které chtějí mít e-maily pod vlastní správou.
2. **Flexibilita:**
    - Možnost přizpůsobit nastavení podle vlastních potřeb (velikost schránek, spam filtry).
3. **Bezpečnost:**
    - Lepší ochrana dat a možnost implementace vlastních bezpečnostních opatření.
4. **Identita:**
    - Vlastní doména zvyšuje profesionální vzhled (např. jmeno@mojefirma.cz).

Mailserver je klíčový pro odesílání a přijímání e-mailů, umožňuje vlastní správu poštovních schránek a nabízí flexibilitu a bezpečnost. Protokoly SMTP, POP3 a IMAP zajišťují komunikaci mezi klienty a servery. Nastavení a správa mailserveru vyžadují základní znalosti konfigurace serveru, ale výhody, jako je kontrola nad daty a profesionální e-mailová komunikace, stojí za investovaný čas.

16. **crond: co to je za daemona, jak se plní, k čemu slouží, uveď ideální použití**

Crond je systémový démon v Unixových a Linuxových systémech, který spravuje a vykonává naplánované úlohy v určených časových intervalech. Funguje na základě definic v souborech crontab.

- **Daemon:**
    - Běží neustále na pozadí.
    - Pravidelně kontroluje naplánované úlohy a vykonává je, pokud jejich čas nastal.

Jak se plní (konfigurace crontab)

Crontab je soubor obsahující seznam naplánovaných úloh a jejich časové definice. Crontab lze konfigurovat pro jednotlivé uživatele nebo na úrovni systému.

Struktura crontab záznamu:

Každý řádek má následující formát:

min hod den měsíc den_v_týdnu příkaz

- **min**: Minuta (0–59)
- **hod**: Hodina (0–23)
- **den**: Den v měsíci (1–31)
- **měsíc**: Měsíc (1–12)
- **den_v_týdnu**: Den v týdnu (0–7, kde 0 a 7 jsou neděle)

Příklady:

- **Každý den ve 3:00 ráno:**

```
0 3 * * * /path/to/script.sh
```

- **Každé pondělí ve 12:00:**

```
0 12 * * 1 /path/to/backup.sh
```

-

Globální konfigurace (pro celý systém):

- Systémové crontab soubory se nachází v /etc/crontab nebo /etc/cron.d/

K čemu slouží

Crond je ideální pro automatizaci opakujících se úloh. Mezi hlavní případy použití patří:

1. **Zálohování dat:**
    - Automatizace záloh na denní nebo týdenní bázi.
2. **Údržba systému:**
    - Rotace logů, čištění dočasných souborů, restartování služeb.
3. **Monitorování:**
    - Spouštění skriptů pro kontrolu dostupnosti serverů nebo systémových zdrojů.
4. **Automatizace aplikací:**
    - Spouštění skriptů, které provádějí údržbu nebo aktualizaci databází.

Ideální použití

1. Zálohování databáze
2. Čištění dočasných souborů
3. Kontrola serveru
4. Rotace logů - Automatická správa logovacích souborů

Bezpečnostní úvahy

1. **Omezení přístupu:**
    - Crontab lze spravovat pouze oprávněnými uživateli. Povolení/zakázání uživatelů spravujete v souborech:
        - **/etc/cron.allow**
        - **/etc/cron.deny**
2. **Logování výstupů:**
    - Výstupy z úloh jsou obvykle odesílány e-mailem uživateli. Alternativně lze logovat do souboru:
3. **Testování skriptů:**

- Před zařazením skriptu do crontab je důležité otestovat jeho funkčnost ručně.

Crond je klíčový démon pro automatizaci úloh v Linuxu. Umožňuje pravidelně spouštět příkazy nebo skripty podle definovaných časových rozvrhů. Ideální je pro zálohování, údržbu systému, monitorování a další opakující se úkoly. Správné nastavení a logování zajišťují efektivní a bezpečné používání tohoto nástroje.

17. **Docker: proč je tak oblíbený, co přináší, pro koho je to dobré**

Docker je platforma pro vytváření, distribuci a běh kontejnerů, které umožňují izolovat aplikace a jejich závislosti v jednotném prostředí. Jeho popularita pramení z následujících výhod:

1. **Jednoduchost a rychlost:**
    - Docker umožňuje snadno vytvářet a nasazovat aplikace díky použití kontejnerů, které jsou lehčí a rychlejší než tradiční virtuální stroje.
    - Startování kontejnerů je téměř okamžité.
2. **Izolace:**
    - Každý kontejner běží nezávisle a obsahuje všechny potřebné závislosti, což minimalizuje konflikty mezi aplikacemi na stejném serveru.
3. **Portabilita:**
    - Kontejnery běží stejným způsobem na různých prostředích (vývojové počítače, testovací servery, produkční infrastruktura).
4. **Efektivita zdrojů:**
    - Kontejnery sdílejí jádro operačního systému, což znamená nižší nároky na paměť a procesor oproti virtuálním strojům.
5. **Automatizace a škálovatelnost:**
    - Docker podporuje automatizaci nasazení, škálování aplikací a snadnou správu prostředí.

Co Docker přináší

1. **Kontejnerizace aplikací:**
    - Aplikace jsou baleny do samostatných kontejnerů, které obsahují všechny závislosti (např. knihovny, binární soubory).
2. **Jednotné prostředí:**
    - Vývojáři mohou pracovat na stejném prostředí, jaké se používá v produkci, což snižuje riziko problémů typu "funguje to u mě, ale ne na serveru".
3. **Ekosystém:**
    - Docker nabízí širokou škálu nástrojů, jako je Docker Compose (pro orchestraci více kontejnerů) nebo Docker Hub (repozitář předpřipravených obrazů).
4. **Integrace s DevOps:**
    - Docker se stal standardem v DevOps procesech, protože zjednodušuje CI/CD (Continuous Integration/Continuous Deployment).
5. **Rychlé nasazení:**
    - Pomocí předpřipravených obrazů lze snadno a rychle nasazovat složité aplikace.

Pro koho je Docker dobrý

1. **Vývojáři:**
    - Docker umožňuje rychle nastavit vývojové prostředí bez složité instalace závislostí.
    - Zaručuje, že prostředí vývoje odpovídá produkčnímu prostředí.
2. **Správci systémů:**
    - Docker usnadňuje nasazení a správu aplikací díky izolaci a portabilitě kontejnerů.
    - Umožňuje snadno spravovat více aplikací na jednom serveru.

3. **Týmy DevOps:**
    - Docker podporuje CI/CD procesy a zjednodušuje škálování aplikací.
    - Umožňuje rychle testovat a nasazovat nové verze aplikací.
4. **Firmy s cloudovou infrastrukturou:**
    - Kontejnery jsou ideální pro nasazení v cloudu díky jejich lehkosti a přenositelnosti.
5. **Startupy:**
    - Umožňuje rychlé prototypování a nasazení aplikací bez potřeby velkých investic do infrastruktury.

Příklady použití

1. **Mikroservisní architektura:**
    - Každá služba běží v samostatném kontejneru, což usnadňuje škálování a správu.
2. **Testovací prostředí:**
    - Docker umožňuje rychle vytvořit izolované testovací prostředí pro nové verze aplikací.
3. **Vývoj a nasazení webových aplikací:**
    - Nástroje jako Docker Compose umožňují jednoduše spouštět aplikace složené z více komponent (např. databáze, backend, frontend).
4. **Migrace mezi servery:**
    - Kontejnery lze snadno přesouvat mezi různými servery nebo cloudovými poskytovateli.

Docker je oblíbený díky své jednoduchosti, rychlosti, portabilitě a efektivnímu využívání zdrojů. Je ideální pro vývoj, testování i produkční nasazení aplikací. Umožňuje moderním týmům rychle přecházet z vývoje k nasazení díky jednotnému prostředí a podpoře automatizace. Docker je skvělým nástrojem pro firmy všech velikostí, které chtějí zefektivnit práci s aplikacemi a optimalizovat svou infrastrukturu.

18. **bezpečnostní audit systému: na co se zaměřit, jaké nástroje použít, co si z toho odnést**

Bezpečnostní audit systému je proces hodnocení a analýzy zabezpečení systému za účelem odhalení zranitelností a rizik. Je to klíčová aktivita pro ochranu dat, minimalizaci hrozeb a splnění bezpečnostních standardů.

Na co se zaměřit

1. **Správa uživatelů a oprávnění:**
    - Kontrola uživatelských účtů:
        - Zda existují neaktivní účty nebo účty s výchozími hesly.
        - Zda má každý uživatel jen ta oprávnění, která potřebuje.
    - Kontrola skupin a jejich práv.
    - Sledování přihlášení (logy, neúspěšné pokusy o přihlášení).
2. **Aktualizace systému:**
    - Zajistit, že všechny balíky a jádro systému jsou aktuální a neobsahují známé zranitelnosti.
    - Kontrola pravidelných bezpečnostních aktualizací.
3. **Nastavení firewallu:**
    - Zda jsou povoleny pouze nezbytné porty a služby.
    - Testování firewallu vůči útokům (např. port scanning).
4. **Služby běžící na systému:**
    - Kontrola spuštěných služeb a procesů.
    - Identifikace nepotřebných nebo zranitelných služeb.
5. **Logovací systém:**
    - Zajištění, že logy jsou správně ukládány, rotovány a monitorovány.
    - Analýza logů na podezřelé aktivity.
6. **Šifrování:**
    - Použití šifrování pro ukládání citlivých dat (disk, databáze).
    - Kontrola, zda jsou přenosy dat šifrovány (TLS/SSL).
7. **Zranitelnosti:**
    - Testování systému vůči známým zranitelnostem.
    - Kontrola konfigurace proti bezpečnostním best practices.
8. **Fyzická bezpečnost:**
    - Přístup k serverům, zálohy, hardware

Jaké nástroje použít

1. **Automatizované skenery:**
    - **Lynis:** Bezpečnostní audit Linuxových a Unixových systémů.
    - **OpenVAS:** Open-source nástroj pro analýzu zranitelností.
    - **Nessus:** Komplexní skener zranitelností.
    - **Nikto:** Skenování webových serverů na zranitelnosti.
2. **Monitorovací nástroje:**
    - **Auditd:** Pro sledování aktivit uživatelů a přístupů k souborům.

    - **Syslog:** Centrální logovací systém.
3. **Testování firewallu:**
    - **Nmap:** Mapování sítě a detekce otevřených portů.
    - **iptables-log:** Pro sledování pravidel firewallu.
4. **Kontrola aktualizací:**
    - **Unattended-upgrades:** Automatické bezpečnostní aktualizace v Debian/Ubuntu.
    - **Yum/DNF security plugin:** Kontrola bezpečnostních aktualizací v RedHat/CentOS.
5. **Šifrování a konfigurace:**
    - **OpenSSL:** Kontrola SSL/TLS certifikátů a konfigurace.
    - **Testssl.sh:** Analýza síťového šifrování.
6. **Analýza logů:**
    - **Logwatch:** Automatizovaná analýza logů.
    - **Splunk:** Pokročilé zpracování a analýza logů.
7. **Hardening:**
    - **Bastille Linux:** Nástroj pro bezpečnostní hardening systému.
    - **SELinux/AppArmor:** Omezení přístupu na úrovni jádra.

Co si z toho odnést

1. **Identifikace zranitelností:**
    - Seznam konkrétních problémů, jako jsou zastaralé balíky, nechráněné porty nebo nesprávně nastavené služby.
2. **Prioritizace oprav:**
    - Posouzení závažnosti zjištěných problémů a vytvoření plánu pro jejich nápravu.
3. **Bezpečnostní doporučení:**
    - Implementace bezpečnostních politik a postupů (např. pravidelné audity, monitoring).
4. **Zlepšení zabezpečení:**
    - Lepší ochrana před útoky, snížení rizika ztráty dat a větší důvěra v systém.
5. **Shoda s předpisy:**
    - Zajištění souladu se zákony a standardy (např. GDPR, ISO 27001).

deální použití bezpečnostního auditu

1. **Pravidelný audit:**
    - Provádět pravidelné kontroly alespoň jednou za čtvrtletí.
2. **Před nasazením nové aplikace:**
    - Ověřit, že nový systém splňuje bezpečnostní požadavky.
3. **Po bezpečnostním incidentu:**
    - Zjistit příčinu a zabránit opakování problému.
4. **Při změně infras**
5. **truktury:**

- Po migraci na nový server, přidání nových služeb nebo změně konfigurace.

Bezpečnostní audit není jednorázová akce, ale kontinuální proces, který zajišťuje aktuální a efektivní ochranu systému.