A DevOps Engineer's Journey

`Author: Kasra Jamshidi`

Github

HTTP
HTTP (HTTP status cods)
Set IP
NGINX
CA
CERTBOT (CA)
DNS
ACL
JAIL (BIND)
LVM
NFS
TTFB
FIREWALL
ICMP(echo)
NAMESPACES
DOCKER

To avoid confusion, since I'm using the history of my server, my first server's IP is 192.168.69.54 and the second one is 192.168.69.55.

"This repository contains personal notes and resources related to DevOps. It is not intended for production use, and should not be relied upon for critical tasks. Use at your own discretion."

HTTP (Hypertext Transfer Protocol) :

یکی از پروتکل‌های اصلی در وب است که برای انتقال اطلاعات بین کلاینت (معمولاً مرورگر) و سرور استفاده می شود .

اجزای اصلی HTTP

HTTP Request (درخواست)

درخواست HTTP از سمت کلاینت به سرور ارسال می‌شود و شامل دو بخش اصلی است:

1. **Header**: اطلاعات متا درباره درخواست، مثل نوع درخواست (GET، POST، و غیره)، نوع محتوا، و غیره.
2. **Body**: داده‌هایی که همراه درخواست ارسال می‌شوند (معمولاً در POST یا PUT).

HTTP Response (پاسخ)

پاسخ HTTP از سمت سرور به کلاینت ارسال می‌شود و شامل دو بخش اصلی است:

1. **Header**: اطلاعات متا درباره پاسخ، مثل نوع محتوا و وضعیت درخواست.
2. **Body**: محتوای واقعی پاسخ، مثل HTML، JSON، یا فایل.

روش‌های HTTP (HTTP Methods)

برای تعریف نوع درخواست کلاینت به سرور استفاده می‌شوند:

متد	کاربرد
GET	درخواست برای دریافت یک منبع مشخص بدون تغییر در سرور.
POST	ارسال داده به سرور برای پردازش (معمولاً برای ارسال فرم).
PUT	ارسال داده برای ذخیره در یک مکان مشخص.
DELETE	حذف یک منبع مشخص روی سرور.
HEAD	مشابه GET، اما فقط هدر را بدون بدنه پاسخ می‌گیرد.
OPTIONS	اطلاعات درباره متدهای پشتیبانی شده برای یک منبع خاص را برمی‌گرداند.
TRACE	برای آزمایش و اشکال‌زدایی مسیر درخواست به سرور استفاده می‌شود.

کدهای وضعیت HTTP (HTTP Status Codes)

کدهای وضعیت در پاسخ HTTP نشان‌دهنده وضعیت پردازش درخواست هستند. این کدها به چند دسته اصلی تقسیم می‌شوند:

دسته	معنی
1XX	اطلاعاتی (در حال پردازش درخواست)
2XX	موفقیت (درخواست با موفقیت پردازش شد)
3XX	تغییر مسیر (ریدایرکت به یک آدرس دیگر)
4XX	خطای کلاینت (مشکل در درخواست کلاینت)
5XX	خطای سرور (مشکل در سرور در پردازش درخواست)

کدهای مهم HTTP Status

دسته 1XX: Informational

- **100 Continue**: سرور آماده دریافت ادامه درخواست است.

- **101 Switching Protocols**: این کد نشان می‌دهد که سرور درخواست کلاینت برای تغییر پروتکل را پذیرفته است. به‌عنوان مثال، اگر کلاینت بخواهد از HTTP به WebSocket تغییر کند، این کد ارسال می‌شود.
- **102 Processing** (WebDAV): این کد نشان می‌دهد که سرور در حال پردازش درخواست است، اما پاسخ نهایی هنوز آماده نیست. معمولاً در عملیات‌های طولانی‌مدت در WebDAV استفاده می‌شود.

دسته 2XX: Success

- **200 OK**: درخواست با موفقیت انجام شد.
- **201 Created**: منبع جدیدی ایجاد شده است.
- **202 Accepted**: این کد به این معنی است که درخواست دریافت شده و در حال پردازش است، اما هنوز تکمیل نشده است.
- **204 No Content**: این کد نشان می‌دهد که درخواست با موفقیت انجام شده، اما پاسخی از سمت سرور برنمی‌گردد (هیچ محتوایی ارسال نمی‌شود). معمولاً در پاسخ به درخواست‌هایی مثل حذف منابع استفاده می‌شود.
- **206 Partial Content**: این کد زمانی استفاده می‌شود که کل محتوا ارسال نشده و فقط بخشی از آن (بر اساس درخواست کاربر) ارسال شده است. به‌طور مثال، در دانلودهای قطع‌شده یا دانلودهای تکه‌ای.

دسته 3XX: Redirect

- **301 Moved Permanently**: منبع به صورت دائمی به مکان جدید منتقل شده است.
- **302 Found (Temporary Redirect)**: منبع به صورت موقت به مکان جدید منتقل شده است.
- 304 **Not Modified**: این کد نشان می‌دهد که منبع مورد درخواست از زمان آخرین کش شدن توسط مرورگر، تغییر نکرده است. در نتیجه، مرورگر می‌تواند نسخه کش‌شده را استفاده کند.
- 307 **Temporary Redirect**: مشابه کد 302 است، اما در اینجا مرورگر باید از همان متد HTTP (مانند POST یا GET) که در درخواست اصلی استفاده شده است، برای درخواست جدید نیز استفاده کند.
- 308 **Permanent Redirect**: مشابه کد 301 است، با این تفاوت که مرورگر ملزم است از همان متد HTTP استفاده کند. این کد نشان‌دهنده انتقال دائمی است.

دسته 4XX: Client Error

- **400 Bad Request**: درخواست نادرست (Syntax مشکل دارد).
- **401 Unauthorized**: احراز هویت لازم است.
- **403 Forbidden**: دسترسی به منبع ممنوع است.
- **404 Not Found**: منبع پیدا نشد.
- 405 Method Not Allowed: متد HTTP استفاده‌شده (مثلاً GET یا POST) برای این منبع مجاز نیست.
- **406 Not Acceptable** : سرور نمی‌تواند پاسخی بر اساس محتوای درخواستی کلاینت ارائه دهد (مثلاً اگر کلاینت فرمت خاصی را درخواست کرده باشد و سرور نتواند آن را تولید کند).
- **408 Request Timeout**: کلاینت برای ارسال درخواست بیش از حد طول کشیده است، بنابراین سرور ارتباط را قطع کرده است.
- **409 Conflict**: این کد نشان می‌دهد که در اجرای درخواست یک تضاد وجود دارد، مثلاً هنگام تلاش برای به‌روزرسانی یک منبع که هم‌زمان توسط درخواست دیگری تغییر کرده است.

- **410 Gone**:این منبع موردنظر دیگر در سرور موجود نیست و احتمالاً به صورت دائمی حذف شده است. برخلاف 404، کد نشان می‌دهد که حذف منبع آگاهانه بوده است.

دسته 5XX: Server Error

- **500 Internal Server Error**: خطای داخلی سرور.چ
- **501 Not Implemented**: سرور توانایی یا قابلیت اجرای متد درخواست‌شده را ندارد. مثلاً اگر کلاینت از متدی استفاده کند که سرور آن را پشتیبانی نمی‌کند
- **502 Bad Gateway**: مشکل در ارتباط با سرور بالادستی
- **503 Service Unavailable**: سرویس در دسترس نیست
- **504 Gateway Timeout**: زمان پاسخ‌دهی سرور بالادستی تمام شده است
- **505 HTTP Version Not Supported**: سرور نسخه HTTP استفاده‌شده در درخواست کلاینت را پشتیبانی نمی‌کند.

ساختار HTTP Request

```
GET /example HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0
Accept: text/html,application/xhtml+xml
```

توضیحات:

1. خط اول: متد، آدرس منبع، و نسخه پروتکل.
2. هدرها: اطلاعاتی مثل نوع مرورگر (User-Agent) و نوع داده مورد پذیرش.

ساختار HTTP Response

```
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 125

<html>
<body>
<h1>Success!</h1>
</body>
</html>
```

توضیحات:

1. خط اول: نسخه پروتکل، کد وضعیت، و توضیح وضعیت.
2. هدرها: نوع داده و اندازه محتوا.
3. بدنه: محتوای واقعی پاسخ.

تنظیم IP

as a webserver
as a reverse proxy
as a loadbalance

بر روی ماشین مجازی :

1. set IP Address
2. change password
3. Install bash-completion

   username : root password ==> روی ماشین ها کلیک کرده
4. IPa : چک کردن IP روی ماشین
5. puTTY : باز کردن پوتی و وارد کردن IP مورد نظر
6. username : root password : -
7. Redhat ==> vi ete/ sysconfig/Network-scripts/
8. Systemctl restart Netwark Manager
9. iptables -F

ifcfg.ens32 ==> space نمایش کانفیگ کارت شبکه با زدن

تنظیم دستی IP

```
| BOOTPROTO = None or Static
| IPADDR = مورد نظر ip
| MASK = 255.255.255.0
| GATEWAY = 192.168.67.254
| ONBOOT = YES
```

اگر server ری استارت شود تنظیمات شبکه UP می شود .

راه دوم

1. nmtui
2. Edit a connection
3. Edit
4. IPV4 { Address , Gateway , DNS }

5. Aatomatically connect
6. Back
7. Active

مرحله دوم : ssh زدن به سرور
مرحله سوم : تغییر دادن پسورد
مرحله چهارم : `dnf -y install vim bash-completion`

NGINX

یک وب سرور است ولی عموما به عنوان یک reverse proxy استفاده می شود . :NIGNX

- نباید nignx رو از ریپوز دیفالت دانلود کرد چون ممکن است نسخه قدیمی و دچار باگ امنیتی باشد .

https://nginx.org/en/linux_packages.html

روی NGINX می‌توانیم برنامه‌های نوشته‌شده با جاوا و .NET را تنظیم و اجرا کنیم. برای نصب، باید ریپوزیتوری مرتبط با آن را اضافه کنیم.

نصب NGINX:

مراحل نصب در RedHat:

ورود به دایرکتوری مخازن :

```
cd /etc/yum.repos.d/
```

ویرایش فایل مخزن:

```
vim nginx.repo
```

تغییر نام فایل مخزن قدیمی:

```
mv rabbitmq.local.repo rabbitmq.local.bak
```

نصب NGINX:

```
yum -y install nginx
```

فعال‌سازی و شروع سرویس NGINX :

```
systemctl enable --now nginx
```

پاک کردن قوانین فایروال:

```
iptables -F
```

غیرفعال کردن سرویس firewalld:

```
systemctl disable --now firewalld
```

دایرکتوری ریشه پیش‌فرض NGINX:

دایرکتوری ریشه‌ی NGINX (جایی که محتوای وب‌سایت ذخیره می‌شود):

1. ورود به دایرکتوری NGINX HTML:

```
cd /usr/share/nginx/html/
```

تغییر نام فایل index.html:

```
mv index.html index.html.bak
```

ایجاد یک فایل جدید index.html:

```
echo "hello" > index.html
```

HTTPS:

ما انواع حملات تحت وب داریم که مربوط به هم به انواع مختلف است. آیا داشتن HTTPS در سایت می‌تواند جلوی حملات SQL Injection را بگیرد؟ خیر، این نوع حمله مستقیماً به دیتابیس دسترسی پیدا می‌کند و داده‌ها را تغییر می‌دهد. HTTPS فقط داده‌های منتقل شده را رمزنگاری می‌کند.

ء Hypertext Transfer Protocol Secure** مخفف HTTPS است . این پروتکل داده‌های شما را به صورت رمزنگاری شده جابه‌جا می‌کند، اما به تنهایی نمی‌تواند امنیت کامل سایت را تضمین کند.

رمزنگاری (Encryption):

رمزنگاری به معنای تبدیل داده‌ها به کدهایی است که از دسترسی غیرمجاز جلوگیری می‌کند. الگوریتم‌های مختلفی برای این کار وجود دارد که اطمینان حاصل می‌کند مهاجمان نمی‌توانند داده‌های شما را تغییر داده یا بخوانند.

هش (Hash):

هش توابعی هستند که داده ورودی را به یک رشته ثابت از کاراکترها تبدیل می‌کنند. این امر برای اطمینان از صحت و یکپارچگی داده‌ها استفاده می‌شود.

شرایط یک گواهینامه معتبر (CA):

یک **مرجع صدور گواهینامه** (Certificate Authority یا CA) معتبر باید موارد زیر را تضمین کند:

1. **اعتبار گواهینامه**: اطمینان از این که گواهینامه توسط یک مرجع معتبر صادر شده و به صورت قانونی و قابل اعتماد قابل استفاده است.
2. **تاریخ اعتبار**: بررسی این که گواهینامه همچنان معتبر بوده و از تاریخ انقضا یا شروع خارج نشده است.
3. **نام گواهینامه**: تطابق نام (Subject Name) گواهینامه با مالک یا دامنه‌ای که برای آن صادر شده است.
4. **فهرست ابطال گواهینامه (CRL)**: گواهینامه نباید در لیست ابطال گواهینامه (Certificate Revocation List یا CRL) یا پایگاه داده مشابهی که گواهینامه‌های لغوشده را نگهداری می‌کند، قرار داشته باشد.

HPKP (HTTP Public Key Pinning):

HPKP یک ویژگی امنیتی برای وب‌سایت‌های HTTPS بود که از حملات جعل گواهی‌نامه (Certificate Forgery) جلوگیری می‌کرد. این کار با مشخص کردن یک یا چند کلید عمومی (Public Key) مورد اعتماد برای مرورگر انجام می‌شد.

عملکرد HPKP:

1. سایت یک هدر خاص به نام `Public-Key-Pins` به مرورگر ارسال می‌کرد.
2. این هدر لیستی از کلیدهای عمومی مورد اعتماد سرور را در خود داشت.
3. مرورگر این کلیدها را ذخیره می‌کرد (Pin می‌کرد) و هنگام بازدید دوباره از سایت، بررسی می‌کرد که آیا گواهی دریافتی با کلیدهای مشخص‌شده در لیست مطابقت دارد یا خیر.
4. اگر مطابقت نداشت، مرورگر از برقراری ارتباط جلوگیری می‌کرد تا از حملاتی مانند **Man-in-the-Middle (MITM)** جلوگیری شود.

چرا HPKP دیگر استفاده نمی‌شود؟

- اگر کلیدها اشتباه یا منقضی می‌شدند، سایت ممکن بود برای کاربران غیرقابل دسترس شود.
- مدیریت آن پیچیده بود و خطاهای پیکربندی می‌توانست به مشکلات جدی منجر شود.
- این مکانیزم جای خود را به مکانیزم‌های ساده‌تر و امن‌تری مانند Certificate Transparency و HSTS داده است.

تنظیمات Self-Signed CA و ایجاد Certificate Chain

1. مسیردهی به فایل‌های تنظیمات NGINX

```
cd /etc/nginx/conf.d/
```

2. حذف فایل‌های قبلی (در صورت لزوم)

```
rm -rf *
```

3. ایجاد کلید خصوصی (Private Key)

```
openssl genrsa -des3 -out server.key 2048
```

4. حذف رمز عبور از کلید خصوصی

```
openssl rsa -in server.key -out private.key
```

5. **تبدیل کلید خصوصی برای استفاده توسط CA**
   فایل `private.key` برای استفاده آماده است.
6. **ایجاد CSR (Certificate Signing Request)**
   - اگر نیاز به CSR برای ارسال به CA داشتید، این مرحله انجام می‌شود.
7. **ایجاد گواهی (Certificate)**

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout private.key
```

مشخص کردن اطلاعات گواهی

8. شامل نام، ایمیل، CA و دیگر اطلاعات.

قرار دادن گواهی‌های دیجیتال در سخت‌افزار به جای NGINX یا نرم‌افزارهای دیگر، یک روش بهینه برای افزایش عملکرد و امنیت است. در ادامه دلایل و نحوه انجام این کار توضیح داده شده است:

- **افزایش امنیت:**
  - کلید خصوصی هرگز از سخت‌افزار خارج نمی‌شود، بنابراین امکان سرقت آن کاهش می‌یابد.
  - سخت‌افزارهای اختصاصی مانند HSM (Hardware Security Module) مکانیزم‌های امنیتی داخلی برای جلوگیری از دسترسی غیرمجاز دارند.
- **بهبود عملکرد:**
  - سخت‌افزارهای اختصاصی قادر به پردازش عملیات رمزنگاری (مانند TLS Handshake) با سرعت بسیار بیشتری هستند.
  - کاهش بار CPU سرور اصلی، مخصوصاً در سرورهایی با درخواست‌های رمزنگاری بالا.
- **مدیریت آسان‌تر:**

- سخت‌افزارهای مخصوص گواهی‌ها امکان مدیریت متمرکز کلیدها و گواهی‌ها را فراهم می‌کنند.
- امکان استفاده از گواهی‌ها برای چندین سرور یا برنامه به طور همزمان.

گام‌های گرفتن گواهی از Let's Encrypt

1. نصب Certbot :

سرت بات (Certbot) ابزاری است که به صورت خودکار برای شما گواهی Let's Encrypt دریافت می‌کند. برای نصب آن، از دستورات زیر استفاده کنید:

در CentOS/RHEL:

```
yum install epel-release -y
yum install certbot python3-certbot-apache -y
```

در Ubuntu/Debian:

```
apt update
apt install certbot python3-certbot-apache -y
```

2. درخواست گواهی

بعد از نصب Certbot، کافی است دستور زیر را اجرا کنید:

2. درخواست گواهی

بعد از نصب Certbot، کافی است دستور زیر را اجرا کنید:

```
certbot --apache
```

این دستور:

- دامنه‌های پیکربندی‌شده در **Apache** را شناسایی می‌کند.
- فایل‌های لازم برای گواهی را ایجاد می‌کند.
- گواهی از **Let's Encrypt** دریافت می‌کند.
- سرور را برای استفاده از HTTPS به‌روز می‌کند.

3. اگر سرور Apache ندارید (DNS-based)

اگر از وب سروری غیر از Apache استفاده می‌کنید، یا فقط قصد دارید گواهی بگیرید بدون تغییر سرور :

```
certbot certonly --standalone -d example.com -d www.example.com
```

`-d example.com` : دامنه‌ای که می‌خواهید گواهی برای آن بگیرید.

TIP : این روش نیاز دارد که پورت 80 روی سرور باز باشد

4. تمدید خودکار گواهی

Let's Encrypt گواهی‌های 90 روزه صادر می‌کند، بنابراین باید تمدید خودکار فعال شود. Certbot به صورت پیش‌فرض یک cron job یا systemd timer برای تمدید خودکار ایجاد می‌کند.

برای تست تمدید:

```
certbot renew --dry-run
```

DNS

دی‌ان‌اس (DNS) سیستمی است که وظیفه تبدیل نام دامنه‌ها به آدرس‌های IP را بر عهده دارد تا دستگاه‌ها در شبکه بتوانند با یکدیگر ارتباط برقرار کنند.

انواع سرور DNS:

1. Master Zone (زون اصلی)

این زون مثل دفتر اصلی تلفن است. شما تمام اطلاعات دامنه‌ها و IPها را در اینجا ذخیره می‌کنید.
مثال: فرض کنید شما مالک دامنه `jamshidi.ir` هستید. اطلاعاتی مثل `www.jamshidi.ir` برابر است با `192.168.5.131` را در این زون وارد می‌کنید. این زون مرکز اصلی داده‌های دامنه است.

2. Slave Zone (زون پشتیبان)

این زون کپی زون Master است و اگر Master در دسترس نباشد، از این زون استفاده می‌شود.
مثال: شما یک سرور پشتیبان دارید. اگر سرور اصلی DNS شما خراب شود، سرور Slave می‌تواند به درخواست‌های DNS پاسخ دهد.

3. Forward Zone (زون ارسال‌کننده)

این زون وقتی استفاده می‌شود که سرور DNS شما خودش اطلاعات را ندارد و باید سوال‌ها را به یک سرور دیگر ارسال کند.
مثال: سرور DNS شما نمی‌داند `example.com` کجاست، پس از یک سرور خارجی (مثل Google DNS) می‌پرسد.

4. Hint Zone (زون راهنما)

این زون لیستی از آدرس سرورهای Root DNS است و وقتی استفاده می‌شود که هیچ اطلاعاتی در دسترس نیست.
مثال: سرور DNS شما نمی‌داند از کجا شروع کند، پس به سرورهای Root (مثل `com.` یا `ir.`) مراجعه می‌کند و اطلاعات اولیه را از آنجا می‌گیرد.

اصطلاحات:

- **TLD (Top Level Domain):** سطح اول دامنه (مانند `.com` , `.ir`).
- **FQDN (Fully Qualified Domain Name):** نام کامل دامنه (مانند `mail.cando.com`).
- **Authoritative DNS:** پاسخ‌دهنده اصلی و دقیق به درخواست‌ها.
- **Non-Authoritative DNS:** پاسخ‌هایی که کش شده‌اند و معتبر نیستند.

مفاهیم اولیه:

1. **Recursion:**
   - باید غیر فعال باشد تا سرور DNS توسط بات‌ها مورد پرسش قرار نگیرد.
2. **DNSSEC:**
   - یک پروتکل امنیتی برای تضمین صحت و اعتبار اطلاعات DNS.
3. **TTL (Time to Live):**
   - برای تغییر آدرس IP یک وب‌سایت، مقدار **TTL** باید کاهش یابد (۱ یا ۲ دقیقه) تا تغییرات سریع‌تر اعمال شوند.

کانفیگ فایل‌ها:

نصب DNS Server:

```
yum –y install bind bind–utils
```

شروع سرویس:

```
systemctl enable named
systemctl start named
```

بررسی وضعیت سرویس:

```
systemctl status named
```

فایل تنظیمات اصلی DNS:

```
vim /etc/named.conf
```

فایل اطلاعات zone :

```
vim /var/named/data/jamshidi.db
```

نکات مدیریتی:

1. **قرار دادن سرور اصلی (Master) در DMZ:**
   - ممنوع است، امنیت کاهش می‌یابد.
2. **ساخت سرور Slave:**
   - یک سرور ثانویه تنظیم کنید تا در صورت خرابی سرور اصلی، عملکرد DNS ادامه داشته باشد.
3. **به‌روزرسانی Slave:**
   - از دستور زیر استفاده کنید:

```
rndc reload
```

مقدار **Serial** در فایل زون باید ۱ عدد افزایش یابد.

در فایل **زون** (zone)، مقدار **Serial** یک شماره نسخه برای رکوردهای DNS آن زون است. این مقدار نشان‌دهنده تغییرات در فایل زون است و به سرورهای **Slave** (ثانویه) کمک می‌کند تا تشخیص دهند که آیا فایل زون به‌روزرسانی شده است یا نه.

چرا باید مقدار Serial را افزایش دهیم؟

وقتی تغییری در رکوردهای DNS فایل زون اعمال می‌کنید (مثل افزودن یک رکورد جدید یا ویرایش رکوردهای موجود)، مقدار **Serial** باید **۱ واحد افزایش یابد**. این افزایش ضروری است زیرا سرورهای Slave تنها زمانی فایل زون جدید را از سرور Master می‌گیرند که مقدار Serial در فایل زون جدید، بزرگ‌تر از مقدار فعلی باشد. اگر مقدار Serial را افزایش ندهید:

- سرورهای Slave تصور می‌کنند فایل زون تغییر نکرده است.
- رکوردهای قدیمی در سرورهای Slave باقی می‌مانند.
- درخواست‌ها به‌درستی به سرورهای مقصد هدایت نمی‌شوند.

فرمت مقدار Serial

بهترین روش برای مقداردهی به Serial استفاده از فرمت تاریخ + شماره نسخه است. برای مثال:

- **2024120801**: سال 2024، ماه 12، روز 08، نسخه 01. اگر تغییرات دیگری در همان روز اعمال کنید، شماره نسخه را افزایش می‌دهید: **2024120802**.

نحوه به‌روزرسانی

1. فایل زون را با یک ویرایشگر متنی باز کنید (مثلاً `vim /var/named/data/example.com.db`).
2. مقدار Serial را پیدا کنید.
3. مقدار فعلی را یک عدد افزایش دهید.

4. فایل را ذخیره کرده و از دستور زیر برای بارگذاری مجدد زون استفاده کنید:

```
rndc reload
```

Forwarders:

برای ارسال درخواست‌های DNS به سرورهای دیگر (مثلاً Google):

```
forwarders {
    1.1.1.1;
    8.8.8.8;
};
forward only;
```

فورواردینگ در DNS به این معنی است که وقتی سرور DNS نتواند به سوال شما جواب بدهد، این سوال را به یک سرور DNS دیگر که مشخص کرده‌ایم، ارسال می‌کند تا پاسخ را از آنجا دریافت کند.

نمونه فایل کانفیگ DNS

```
options {
    listen-on port 53 { 192.168.69.54; };
    listen-on-v6 port 53 { ::1; };
    directory       "/var/named";
    dump-file       "/var/named/data/cache_dump.db";
    statistics-file "/var/named/data/named_stats.txt";
    memstatistics-file "/var/named/data/named_mem_stats.txt";
    secroots-file   "/var/named/data/named.secroots";
    recursing-file  "/var/named/data/named.recursing";
    allow-query     { any; };
    recursion no;
    dnssec-enable yes;
    dnssec-validation yes;
```

```
    managed-keys-directory "/var/named/dynamic";
        pid-file "/run/named/named.pid";
        session-keyfile "/run/named/session.key";
    };
    logging {
        channel default_debug {
            file "data/named.run";
            severity dynamic;
        };
    };
    zone "jamshidi.ir" IN {
        type master;
        file "/var/named/data/jamshidi.db";
        allow-update { none ;};
        allow-transfer { 192.168.69.55; };
    };
    zone "." IN {
        type hint;
        file "named.ca";
    };
```

توضیحات فایل کانفیگ DNS

1. بخش options

بخش `options` در فایل تنظیمات DNS شامل تعداد زیادی جزئیات است، اما مهم‌ترین تنظیمات آن عبارت‌اند از:

1. `listen-on port 53 { 192.168.69.54; };`
   مشخص می‌کند سرور فقط به درخواست‌های ارسالی به آدرس **192.168.69.54** روی پورت 53 پاسخ می‌دهد.)پورت پیش‌فرض (DNS
2. `allow-query { any; };`
   تعیین می‌کند که تمام کلاینت‌ها مجاز به ارسال درخواست به سرور هستند. این گزینه برای دسترسی عمومی مهم است.
3. `recursion no;`
   غیرفعال کردن **recursion**، به این معنی که سرور فقط به سوالاتی که مسئولیت آن‌ها را بر عهده دارد (authoritative) پاسخ می‌دهد و سوالات دیگر را حل نمی‌کند.
4. `dnssec-enable yes;`
   فعال کردن **DNSSEC** برای اطمینان از امنیت در پاسخ‌های DNS.
5. `dnssec-validation yes;`

   فعال کردن اعتبارسنجی **DNSSEC** برای بررسی امضاهای دیجیتالی و جلوگیری از جعل DNS.
6. `directory "/var/named";`
   مسیر اصلی ذخیره فایل‌های پیکربندی و زون‌های DNS.

این موارد، مهم‌ترین بخش‌های تنظیم `options` هستند که بر عملکرد سرور تأثیر مستقیم دارند. تنظیمات دیگر برای اهداف خاص یا جزئیات اضافی استفاده می‌شوند.

2. بخش Logging

```
logging {
        channel default_debug {
                file "data/named.run";
                severity dynamic;
        };
};
```

این بخش مربوط به تنظیمات لاگ‌ها (ثبت وقایع) است : logging

یک کانال به نام default_debug تعریف شده که لاگ‌ها را در فایل مشخصی ذخیره می‌کند. : channel default_debug

مسیر فایل لاگ که اطلاعات لاگ به آن نوشته می‌شود. این فایل معمولاً برای خطایابی و دیباگ استفاده می‌شود. : file "data/named.run"

سطح شدت لاگ‌ها را مشخص می‌کند. مقدار dynamic باعث می‌شود شدت لاگ‌ها به صورت خودکار بسته به نیاز تغییر کند. : severity dynamic

3. تعریف زون Master

```
zone "jamshidi.ir" IN {
        type master;
        file "/var/named/data/jamshidi.db";
        allow-update { none; };
        allow-transfer { 192.168.69.55; };
};
```

این بخش مربوط به تعریف زون دامنه jamshidi.ir است. دامنه تعریف‌شده نوع Master است. : zone "jamshidi.ir" IN

مشخص می‌کند این زون نوع Master (سرور اصلی) است که اطلاعات معتبر دامنه را ذخیره و مدیریت می‌کند. : type master

مسیر فایل مربوط به این زون است. این فایل شامل رکوردهای DNS (مانند A، NS و غیره) برای دامنه jamshidi.ir است. : file "/var/named/data/jamshidi.db"

مشخص می‌کند که چه آدرس‌هایی اجازه دارند رکوردهای DNS را در یک زون به‌صورت دینامیک تغییر دهند. : allow-update { none; }

```
allow-update { none; };
```

یعنی هیچ‌کس اجازه تغییر ندارد. این حالت معمولاً برای امنیت بیشتر استفاده می‌شود.

مشخص می‌کند فقط آدرس IP مشخص‌شده (در اینجا: 192.168.69.55) اجازه دریافت اطلاعات زون را دارد. این معمولاً برای سرورهای Slave تنظیم می‌شود. : allow-transfer { 192.168.69.55; }

از ابزارهایی مانند **dig** یا **host** می‌توان برای Zone Transfer استفاده کرد. نمونه‌ای از دستور با **dig**:

```
dig axfr @<DNS_Server_IP> <domain_name>
```

چگونه از این مشکل جلوگیری کنیم؟

برای جلوگیری از Zone Transfer به افراد غیرمجاز:

1. در فایل تنظیمات Zone، فقط به سرورهای معتبر اجازه Zone Transfer بدهید:

```
zone "jamshidi.ir" {
    type master;
    file "/var/named/data/jamshidi.db";
    allow-transfer { 192.168.69.55; };  # فقط به سرور Slave اجازه داده شده
```

```
};
```

اگر Zone Transfer به‌درستی محدود شود، تلاش برای استخراج اطلاعات دامنه با خطا مواجه خواهد شد.
این کار برای جلوگیری از لو رفتن رکورد های dns استفاده می شود .

4. زون Root یا Hint

```
zone "." IN {
        type hint;
        file "named.ca";
};
```

zone "." :

زون `.` به دامنه Root مربوط است و برای راه‌اندازی اولیه سرور DNS استفاده می‌شود.

- **type hint**: به سرور DNS می‌گوید که به سرورهای Root مراجعه کند تا اطلاعات مربوط به دامنه‌های دیگر را پیدا کند.
- **file "named.ca"**: این فایل لیستی از آدرس‌های IP سرورهای Root DNS را دارد و به سرور کمک می‌کند درخواست‌ها را به این سرورها ارسال کند.

ساده‌تر:

این بخش به سرور DNS یاد می‌دهد از کجا شروع کند و اگر چیزی نداند، سراغ سرورهای اصلی (Root) برود.
فایل `named.ca` هم آدرس سرورهای Root را به سرور می‌دهد.

ACL

در تنظیمات DNS، از **ACL** (Access Control List) برای مدیریت دسترسی‌ها استفاده می‌کنیم. یک نمونه ساده از **ACL** برای سرور BIND می‌تواند به این شکل باشد:

```
acl "trusted" {
    192.168.1.0/24;    // Internal network
    10.0.0.0/16;       // Another private network
    localhost;         // The server itself
    127.0.0.1;         // Loopback
};
```

```
options {

allow-query { trusted; };

allow-recursion { trusted; };

allow-transfer { none; };

};
```

این بخش از پیکربندی DNS فقط به کلاینت‌های موجود در "ACL "trusted اجازه می‌دهد که پرس‌وجو (query) و درخواست‌های بازگشتی (recursion) انجام دهند و انتقال زون‌ها (zone transfers) را غیرفعال می‌کند.

DNS Zone File

```
$TTL    604800

@       IN      SOA     ns1.jamshidi.ir. admin.jamshidi.ir. (

                              5         ; Serial

                         604800         ; Refresh

                          86400         ; Retry

                        2419200         ; Expire

                         604800 )       ; Negative Cache TTL

jamshidi.ir.    IN      NS      ns1.jamshidi.ir.
jamshidi.ir.    IN      NS      ns2.jamshidi.ir.

ns1             IN      A       192.168.69.54
ns2             IN      A       192.168.69.54

@               IN      A       192.168.5.129
www             IN      A       192.168.5.131
```

این فایل مربوط به تنظیمات **DNS Zone File** برای دامنه `jamshidi.ir` است که ساختار و اطلاعات اصلی دامنه را مشخص می‌کند. هر بخش را به‌صورت جداگانه توضیح می‌دهم:

1. $TTL 604800

این خط مقدار **Time-To-Live (TTL)** پیش‌فرض رکوردهای DNS را تنظیم می‌کند. مقدار `604800` برحسب ثانیه است (معادل 7 روز). این عدد مشخص می‌کند که اطلاعات DNS چه مدت در کش (Cache) کلاینت‌ها یا سرورهای DNS ذخیره شود.

2. SOA Record (Start of Authority)

رکورد **SOA** اطلاعات اصلی در مورد دامنه و سرور اصلی DNS را ارائه می‌دهد:

- **ns1.jamshidi.ir.**: نام سرور اصلی که مدیریت این دامنه را برعهده دارد
- **admin.jamshidi.ir.**: آدرس ایمیل مدیر دامنه (نقطه‌ها به‌جای @ هستند)
- **Serial (5)**: شماره سریال که هر بار تغییر در فایل اعمال می‌شود، باید افزایش یابد. سرورهای ثانویه از این مقدار برای تشخیص تغییرات استفاده می‌کنند
- **Refresh (604800)**: مدت‌زمانی که سرور ثانویه صبر می‌کند تا تنظیمات سرور اصلی را بررسی کند (7 روز).
- **Retry (86400)**: اگر ارتباط با سرور اصلی قطع شد، بعد از این مدت تلاش مجدد انجام می‌شود (1 روز).
- **Expire (2419200)**: مدت‌زمانی که پس از قطع ارتباط، سرور ثانویه اطلاعات را معتبر نگه می‌دارد (28 روز).
- **Negative Cache TTL (604800)**: مدت‌زمان ذخیره پاسخ‌های منفی (یعنی عدم وجود رکورد) در کش (7 روز).

3. NS Records (Name Server Records)

- این بخش مشخص می‌کند که کدام سرورها به‌عنوان **Name Server (NS)** برای این دامنه عمل می‌کنند:
- `jamshidi.ir. IN NS ns1.jamshidi.ir.`
- `jamshidi.ir. IN NS ns2.jamshidi.ir.`

این دو رکورد به کاربران نشان می‌دهند که برای جستجوی اطلاعات دامنه، باید به سرور `ns1` یا `ns2` مراجعه کنند.

4. A Records (Address Records)

- **ns1 IN A 192.168.69.54**: این رکورد مشخص می‌کند که آدرس IP سرور ns1 برابر با 192.168.69.54 است.
- **ns2 IN A 192.168.69.54**: آدرس سرور ns2 هم با ns1 یکسان است.
- **@ IN A 192.168.5.129**: این رکورد نشان می‌دهد که آدرس IP اصلی دامنه jamshidi.ir برابر با 192.168.5.129 است.
- **www IN A 192.168.5.131**: زیر دامنه www.jamshidi.ir به آدرس 192.168.5.131 اشاره می‌کند.

کاربرد هر بخش به‌طور خلاصه

1. **$TTL**: مدت‌زمان کش شدن اطلاعات
2. **SOA**: مشخصات اصلی سرور و اطلاعات به‌روزرسانی DNS
3. **NS**: سرورهای DNS اصلی و ثانویه برای دامنه
4. **A Records**: تبدیل نام دامنه و زیر دامنه‌ها به آدرس IP

Jailing BIND

یک تکنیک امنیتی که به منظور محدود کردن دسترسی و عملکرد نرم‌افزار BIND به استفاده می‌شود . هدف از این کار این است که حتی اگر مهاجم توانست به سیستم شما حمله کند و سرور DNS را نفوذ کند، توانایی دسترسی به سایر بخش‌های سیستم یا سرور را نداشته باشد. به عبارت دیگر، این کار با ایجاد محدودیت‌های بیشتر در فضای کاری BIND، از آسیب‌های احتمالی جلوگیری می‌کند.

چگونه Jailing BIND کار می‌کند؟
زمانی که BIND در حالت Jail قرار دارد، برنامه BIND به یک دایرکتوری یا محیط محدود دسترسی پیدا می‌کند. در این محیط محدود، BIND تنها به فایل‌ها و منابعی که در این دایرکتوری قرار دارند دسترسی خواهد داشت و از دسترسی به سایر بخش‌های سیستم جلوگیری می‌شود.

اینکار مشابه به یک sandbox است که در آن برنامه نمی‌تواند خارج از محیطی که برایش تعریف شده است، عمل کند.

برای **Jailing BIND** یا محدود کردن آن به یک محیط ایزوله، چندین روش وجود دارد. یکی از رایج‌ترین روش‌ها استفاده از ابزارهایی مانند `chroot` برای قرار دادن BIND در یک دایرکتوری خاص است. در اینجا، به مراحل انجام این کار خواهیم پرداخت.

1. استفاده از chroot برای Jail کردن BIND

chroot (change root) یک دستور در لینوکس است که به شما این امکان را می‌دهد که محیط کاری برنامه‌ها را به یک دایرکتوری خاص محدود کنید. این به این معناست که BIND نمی‌تواند به منابع خارج از آن دایرکتوری دسترسی پیدا کند.

مرحله 1: نصب BIND و پیکربندی اولیه

اولین قدم این است که BIND را نصب و پیکربندی کنید. اگر هنوز نصب نکردید، از دستور زیر استفاده کنید:

```
sudo apt-get update
sudo apt-get install bind9
```

مرحله 2: ایجاد دایرکتوری Jail برای BIND

در این مرحله، یک دایرکتوری برای Jail کردن BIND ایجاد می‌کنید. فرض کنید می‌خواهیم آن را در `var/named/chroot/` قرار دهیم.

```
sudo mkdir -p /var/named/chroot
sudo mkdir -p /var/named/chroot/var/named
```

مرحله 3: کپی کردن فایل‌های مورد نیاز به دایرکتوری Jail

حالا باید فایل‌های مورد نیاز BIND را به داخل Jail منتقل کنید. این فایل‌ها معمولاً شامل پیکربندی‌های BIND و فایل‌های دامنه هستند.

برای مثال:

```
sudo cp -r /etc/bind /var/named/chroot/etc
sudo cp -r /var/cache/bind /var/named/chroot/var/cache
sudo cp -r /var/named /var/named/chroot/var/named
```

مرحله 4: تغییر فایل پیکربندی BIND برای استفاده از chroot

حالا باید فایل پیکربندی BIND را طوری تغییر دهید که از محیط `chroot` استفاده کند.

1. فایل پیکربندی (`/etc/bind/named.conf`) BIND را ویرایش کنید و اطمینان حاصل کنید که مسیرهای مربوط به فایل‌ها و دایرکتوری‌ها به درستی تنظیم شده‌اند. مثلا باید مسیرهایی که به `/var/named/` اشاره دارند، به `var/named/chroot/var/named/` تغییر کنند.
2. همچنین، اگر از `systemd` برای مدیریت BIND استفاده می‌کنید، باید تنظیمات مربوط به آن را برای `chroot` در فایل سرویس BIND تغییر دهید. به طور پیش‌فرض، ممکن است این فایل در `etc/systemd/system/bind9.service/` باشد.

```
[Service]
ExecStartPre=/usr/sbin/chroot /var/named/chroot /bin/bash -c
"/usr/sbin/named -g"
ExecStart=/usr/sbin/named -g
```

مرحله 5: تغییر مالکیت و مجوزهای فایل‌ها

اطمینان حاصل کنید که BIND دسترسی لازم برای فایل‌ها و دایرکتوری‌ها در داخل Jail را دارد. به عنوان مثال:

```
sudo chown -R bind:bind /var/named/chroot
```

مرحله 6: راه‌اندازی مجدد سرویس BIND

بعد از انجام تنظیمات، باید سرویس BIND را دوباره راه‌اندازی کنید تا تغییرات اعمال شوند:

```
sudo systemctl restart bind9
```

LVM

LVM یک سیستم مدیریت دیسک در لینوکس است که به شما اجازه می‌دهد تا دیسک‌های سخت را به‌صورت منطقی مدیریت کنید و حجم‌های منطقی (Logical Volumes) را از فضای دیسک‌های فیزیکی ایجاد کنید. در واقع، LVM به شما انعطاف‌پذیری بیشتری در تخصیص، تغییر اندازه، و مدیریت فضای ذخیره‌سازی می‌دهد.

LVM شامل ۳ جزء اصلی است:

--
Physical Volume (PV)
Volume Group (VG)
Logical Volume (LV)

Physical Extents (PE)

1. Physical Volume (PV)

Physical Volume به هر دیسک یا پارتیشنی گفته می‌شود که در LVM برای ذخیره داده‌ها استفاده می‌شود.

یک PV می‌تواند یک دیسک سخت فیزیکی (مانند `/dev/sda`) یا یک پارتیشن از دیسک (مثل `/dev/sda1`) باشد.

ایجاد PV:
برای استفاده از یک دیسک یا پارتیشن در LVM، ابتدا باید آن را به یک Physical Volume تبدیل کنیم. این کار با دستور `pvcreate` انجام می‌شود.

چرا از PV استفاده می‌شود؟
PVها به‌عنوان اجزای اصلی فضای ذخیره‌سازی در LVM عمل می‌کنند. پس از تبدیل یک دیسک یا پارتیشن به PV، می‌توان آن‌ها را به Volume Group اضافه کرد.

2. Volume Group (VG)

Volume Group مجموعه‌ای از Physical Volumeهاست که به‌طور کلی فضای ذخیره‌سازی را برای Logical Volumeها فراهم می‌کند. یک Volume Group می‌تواند شامل یک یا چند PV باشد و این امکان را فراهم می‌کند که فضای ذخیره‌سازی به‌طور انعطاف‌پذیر مدیریت شود.

ایجاد VG:

بعد از اینکه PVها را آماده کردید، باید آن‌ها را به یک Volume Group اضافه کنید. این کار با دستور `vgcreate` انجام می‌شود.

چرا از VG استفاده می‌شود؟

Volume Group به شما این امکان را می‌دهد که فضای ذخیره‌سازی را به‌طور منطقی و بدون وابستگی به دیسک‌های فیزیکی مدیریت کنید. با افزودن چند PV به یک VG، می‌توانید فضای ذخیره‌سازی بیشتری به‌صورت یکپارچه داشته باشید.

3. Logical Volume (LV)

Logical Volume بخشی از فضای ذخیره‌سازی است که از یک یا چند **Volume Group** استخراج می‌شود. هر Logical Volume به‌طور منطقی مانند یک پارتیشن یا دیسک رفتار می‌کند، اما برخلاف پارتیشن‌های سنتی، می‌تواند تغییر اندازه دهد و فضای بیشتری به آن افزوده شود.

ایجاد LV:

بعد از اینکه یک VG ساخته شد، می‌توانید Logical Volumeها را از آن ایجاد کنید. این کار با دستور `lvcreate` انجام می‌شود.

چرا از LV استفاده می‌شود؟

LVها امکان ایجاد و مدیریت پارتیشن‌های منطقی را با انعطاف‌پذیری بیشتر فراهم می‌کنند. می‌توان آن‌ها را در هر زمانی گسترش داد، کوچک کرد یا حتی حذف کرد. همچنین، می‌توان از آن‌ها برای نصب سیستم‌عامل‌ها، داده‌ها یا فایل‌سیستم‌ها استفاده کرد

+

Physical Extents (PE)

Physical Extent (PE) واحدهای کوچک و ثابت اندازه‌ای هستند که در سطح Physical Volume (PV) به‌کار می‌روند. این‌ها بخش‌هایی از فضای ذخیره‌سازی یک PV هستند که توسط Volume Group (VG) برای تخصیص فضای ذخیره‌سازی به Logical Volumes (LV) استفاده می‌شوند.

هر PE معمولاً به اندازه ۴ مگابایت یا ۸ مگابایت است (می‌توانید این اندازه را در هنگام ایجاد VG تعیین کنید، ولی معمولاً پیش‌فرض ۴ مگابایت است).

PE به‌عنوان واحد تخصیص فضای ذخیره‌سازی عمل می‌کند. زمانی که شما Logical Volume (LV) ایجاد می‌کنید، فضای آن از PEهای موجود در Volume Group گرفته می‌شود.

دستورات کلیدی LVM:

- **`pvcreate`** : برای تبدیل دیسک یا پارتیشن به یک Physical Volume
- **`vgcreate`** : برای ایجاد یک Volume Group جدید از یک یا چند PV
- **`lvcreate`** : برای ایجاد یک Volume Group جدید از یک یا چند PV
- **`lvextend`** : برای گسترش یک Logical Volume
- **`lvreduce`** : برای کوچک کردن حجم یک Logical Volume
- **`vgextend`** : برای افزودن PV جدید به Volume Group.
- **`vgreduce`** : برای حذف PV از یک VG
- **`pvremove`** : برای حذف یک PV از LVM
- **`lvremove`** : برای حذف یک LV
- **`vgremove`** : برای حذف یک VG

نمونه‌ای از نحوه کار LVM:

فرض کنید یک سرور با دو دیسک داریم:

1. `/dev/sdb`
2. `/dev/sdc`

1. شناسایی دیسک‌های جدید

ابتدا دیسک‌ها و پارتیشن‌های موجود را شناسایی کنید:

```
lsblk
lsblk –f
```

اگر دیسک‌ها شناسایی نشده‌اند، دستور زیر را برای اسکن مجدد اجرا کنید:

`

```
echo "– – –" > /sys/class/scsi_host/host0/scan
echo "– – –" > /sys/class/scsi_host/host1/scan
echo "– – –" > /sys/class/scsi_host/host2/scan
```

بعد از اسکن، دوباره دیسک‌ها را بررسی کنید:

```
lsblk –f
```

2. تبدیل دیسک‌ها به Physical Volume

دیسک‌های جدید را به Physical Volume (PV) تبدیل کنید. فرض کنیم دیسک‌های /dev/sdb و /dev/sdc داریم:

```
pvcreate /dev/sdb /dev/sdc
```

3. ایجاد Volume Group

حالا این Physical Volumeها را به یک **Volume Group (VG)** اضافه کنید. مثلاً نام VG را **data** می‌گذاریم:

```
vgcreate data /dev/sdb /dev/sdc
```

4. ایجاد Logical Volume

در این مرحله، یک **Logical Volume (LV)** می‌سازیم. فرض کنیم می‌خواهیم یک LV به نام **lvdata** با حجم 10 گیگابایت ایجاد کنیم:

```
lvcreate -L 10G -n lvdata data
```

5. فرمت کردن Logical Volume

باید LV جدید را فرمت کنید تا قابل استفاده باشد. برای این کار، از فایل سیستم مورد نظر استفاده می‌کنیم:

یا برای سیستم فایل **ext4**:

```
mkfs.ext4 /dev/data/lvdata
```

یا برای سیستم فایل **xfs**:

```
mkfs.xfs /dev/data/lvdata
```

6. مونت (mount) کردن Logical Volume

ابتدا یک دایرکتوری برای mount کردن LV می‌سازیم:

```
mkdir /mnt/lvdata
```

سپس LV را به این مسیر mount می‌کنیم:

```
mount /dev/data/lvdata /mnt/lvdata
```

بررسی تغییرات:
در پایان، با استفاده از دستور زیر مطمئن شوید فضای جدید اعمال شده است:

```
df -h /mnt/lvdata
```

بیشتر کردن فضا (Extend)

1. افزایش اندازه Logical Volume

فرض کنیم می‌خواهید حجم LV را 5 گیگابایت افزایش دهید:

```
lvextend -L +5G /dev/data/lvdata
```

گسترش (Grow) :

با ساختLV بنظر کار به اتمام می رسد . ما فضای Logical Volume را افزایش دادیم، اما سیستم فایل هنوز از این فضای اضافی استفاده نمی‌کند . مانند یک اتاقی که فضای ان را افزایش دادیم ولی صندلی نداریم که بتوان از فضا اضافه شده استفاده کرد .

برای سیستم فایل‌های **ext4**:

```
resize2fs /dev/data/lvdata
```

برای سیستم فایل‌های **xfs**:

```
xfs_growfs /mnt/lvdata
```

بررسی فضای جدید پس از گسترش
پس از گسترش سیستم فایل، فضای جدید را بررسی کنید تا مطمئن شوید اعمال شده است:

```
df –h
```

کم کردن فضا (Reduce)

برای کاهش فضای یک **Logical Volume** در LVM، باید با دقت عمل کنید. این فرآیند شامل مراحل زیر است:

1. اطمینان از استفاده نشدن از فضای Logical Volume

- ابتدا مطمئن شوید که **Logical Volume** در حال استفاده نیست یا داده‌ای که نمی‌خواهید از بین برود، در آن ذخیره نشده است.
- اگر این LV در حال استفاده است، ابتدا آن را **unmount** کنید:

```
umount /mnt/lvdata
```

2. بررسی سلامت سیستم فایل

قبل از کاهش حجم، سیستم فایل باید بررسی و سالم باشد. برای این کار:

```
e2fsck –f /dev/data/lvdata
```

این دستور برای فایل‌سیستم‌های **ext4** است. اگر از **xfs** استفاده می‌کنید، این مرحله نیاز نیست.

3. کاهش اندازه سیستم فایل

ابتدا اندازه سیستم فایل را به مقدار مورد نظر کاهش دهید:

```
resize2fs /dev/data/lvdata 5G
```

(در اینجا حجم جدید را **5 گیگابایت** در نظر گرفته‌ایم.)

⚠️ توجه: حجم جدید باید کمتر یا مساوی حجم نهایی LV باشد.

4. کاهش اندازه Logical Volume

حالا اندازه LV را کاهش دهید:

```
lvreduce -L 5G /dev/data/lvdata
```

(در اینجا حجم جدید LV را **5 گیگابایت** قرار داده‌ایم.)

5. مونت مجدد Logical Volume

بعد از کاهش حجم، LV را دوباره به مسیر مشخص **mount** کنید:

```
mount /dev/data/lvdata /mnt/lvdata
```

6. بررسی فضای نهایی

برای اطمینان از کاهش موفقیت‌آمیز:

```
df -h /mnt/lvdata
```

پشتیبان‌گیری از داده‌ها: کاهش اندازه می‌تواند باعث از دست رفتن داده‌های خارج از محدوده جدید شود. حتماً قبل از انجام این مراحل، از داده‌های خود پشتیبان بگیرید.

NFS

برای راه‌اندازی سرویس **NFS** (Network File System) بر روی یک سیستم لینوکس، ابتدا باید چند مرحله را انجام دهیم که شامل نصب بسته‌های مورد نیاز، تنظیمات لازم برای سرور و کلاینت و سپس تست کردن اتصال بین آن‌ها می‌شود.

1. نصب بسته‌های NFS

برای نصب NFS سرور، دستور زیر را وارد کنید:

```
sudo yum install nfs-utils -y
```

این دستور بسته‌های لازم برای NFS را نصب می‌کند.

2. تنظیمات فایل‌های پیکربندی

برای اینکه دایرکتوری‌ها را به اشتراک بگذارید، باید فایل پیکربندی `etc/exports/` را ویرایش کنید.

ابتدا با دستور `vi` یا `vim` فایل `etc/exports/` را باز کنید:

```
sudo vi /etc/exports
```

سپس دایرکتوری‌ای که می‌خواهید به اشتراک بگذارید را اضافه کنید. برای مثال:

```
/opt/tomcat 192.168.69.0/24(rw,sync,no_root_squash)
```

در فایل پیکربندی (`/etc/exports`) NFS، وقتی یک دایرکتوری را به اشتراک می‌گذارید، می‌توانید پارامترهای مختلفی را برای کنترل دسترسی و نحوه رفتار NFS تنظیم کنید. پارامترهایی مانند `no_root_squash` , `sync` , `rw` که در مثال آمده است، نقش مهمی در تنظیمات امنیتی و دسترسی دارند. در اینجا توضیح می‌دهیم که هرکدام چه کاری انجام می‌دهند:

1. rw (Read-Write)

این گزینه به کلاینت‌ها اجازه می‌دهد که به دایرکتوری اشتراکی هم دسترسی خواندن (Read) و هم نوشتن (Write) داشته باشند.

- **rw**: کلاینت می‌تواند فایل‌ها را هم بخواند و هم تغییر دهد.
- **ro**: اگر از ro استفاده کنید، فقط دسترسی خواندن به کلاینت‌ها داده می‌شود و نمی‌توانند فایل‌ها را تغییر دهند.

2. sync(Synchronous)

پارامتر `sync` مشخص می‌کند که عملیات ورودی/خروجی (I/O) به صورت همزمان انجام شود. این به این معنی است که هر عملیات نوشتن باید تکمیل شود و داده‌ها به دیسک نوشته شوند قبل از اینکه به درخواست بعدی پرداخته شود.

- **sync**: هر عملیات نوشتن منتظر می‌ماند تا داده‌ها به دیسک نوشته شوند و سپس پاسخ به کلاینت داده می‌شود. این باعث اطمینان از پایداری داده‌ها و کاهش ریسک از دست دادن داده‌ها در صورت خاموش شدن غیرمنتظره سیستم می‌شود.
- **async**: اگر از async استفاده کنید، داده‌ها به صورت ناهمزمان نوشته می‌شوند و در نتیجه سرعت دسترسی به فایل‌ها ممکن است بالاتر باشد، اما خطر از دست دادن داده‌ها در صورت وقوع خرابی افزایش می‌یابد.

3. no_root_squash

این گزینه تاثیر زیادی بر رفتار سیستم در زمان استفاده از کاربر root دارد. در NFS، وقتی کلاینت‌ها به سرور متصل می‌شوند، NFS به طور پیش‌فرض کاربران root را به کاربری معمولی تبدیل می‌کند تا از خطرات امنیتی جلوگیری کند. این کار به نام "root squashing" شناخته می‌شود.

- **no_root_squash**: این گزینه باعث می‌شود که کاربر root در کلاینت‌ها همچنان دسترسی‌های root خود را داشته باشد، به این معنی که هیچ تبدیل یا "squash"ای برای کاربر root انجام نمی‌شود.
- **root_squash**: اگر از این گزینه استفاده کنید، کاربر root در کلاینت‌ها به یک کاربر معمولی تبدیل می‌شود (معمولاً nfsnobody)، و بنابراین دسترسی‌های ریشه‌ای از بین می‌روند.

مهم: استفاده از no_root_squash می‌تواند خطر امنیتی داشته باشد، زیرا ممکن است یک کلاینت با دسترسی `root` توانایی انجام عملیات‌های خطرناک روی سرور را داشته باشد. این گزینه معمولاً در محیط‌هایی استفاده می‌شود که کنترل امنیتی بالایی دارند یا نیاز به دسترسی‌های خاصی از سوی root دارند.

جمع‌بندی:

- **rw**: اجازه خواندن و نوشتن به کلاینت‌ها.
- **sync**: اطمینان از نوشتن داده‌ها به دیسک قبل از پاسخ به کلاینت.
- **no_root_squash**: حفظ دسترسی‌های root از طرف کلاینت‌ها (باید با احتیاط استفاده شود).

به طور کلی، شما باید این پارامترها را بسته به نیاز و سطح امنیتی شبکه و سیستم خود تنظیم کنید.

3. راه‌اندازی و فعال‌سازی سرویس NFS

بعد از تنظیمات فایل `/etc/exports`، باید سرویس NFS را راه‌اندازی و فعال کنیم:

```
sudo systemctl start nfs-server
sudo systemctl enable nfs-server
```

برای اطمینان از اینکه سرویس در حال اجرا است، از دستور زیر استفاده کنید:

```
sudo systemctl status nfs-server
```

4. تنظیم فایروال

اگر فایروال فعال باشد، باید پورت‌های NFS را باز کنید. می‌توانید دستور زیر را برای باز کردن پورت‌ها وارد کنید:

```
sudo firewall-cmd --permanent --add-service=nfs
sudo firewall-cmd --permanent --add-service=mountd
sudo firewall-cmd --permanent --add-service=rpc-bind
sudo firewall-cmd --reload
```

5. تنظیمات کلاینت

برای دسترسی به دایرکتوری به اشتراک گذاشته شده از سمت کلاینت، ابتدا باید بسته‌های NFS را نصب کنید:

```
sudo yum install nfs-utils -y
```

سپس دایرکتوری مورد نظر را از سرور مونت کنید:

```
sudo mount -t nfs 192.168.69.54:/opt/tomcat /mnt
```

برای اینکه دایرکتوری به صورت دائمی مانت شود، باید فایل `etc/fstab/` را ویرایش کنید و خط زیر را اضافه کنید:

```
192.168.69.54:/opt/tomcat /mnt nfs defaults 0 0
```

6. تست اتصال

برای اطمینان از عملکرد درست، می‌توانید دستور زیر را برای مشاهده دایرکتوری‌های مانت شده وارد کنید:

```
df -h
```

همچنین می‌توانید از دستور `ls` برای بررسی محتویات دایرکتوری مانت شده استفاده کنید:

```
ls /mnt
```

این مراحل باید سرویس NFS را بر روی سیستم شما راه‌اندازی کرده و اتصال کلاینت به سرور NFS را برقرار کند.

TTFB

یک معیار کلیدی در عملکرد وب‌سایت‌ها و سرورهاست که نشان می‌دهد چه مدت طول **TTFB (Time to First Byte)** می‌کشد تا اولین بایت از داده از سرور به مرورگر کاربر برسد

این زمان شامل ۳ مرحله اصلی است:

1. مدت‌زمان لازم برای یافتن آدرس سرور و برقراری اتصال TCP یا TLS (در صورت استفاده از HTTPS**DNS Lookup : .)
2. **ارسال درخواست:** زمانی که مرورگر درخواست HTTP/HTTPS را به سرور ارسال می‌کند.
3. **دریافت اولین پاسخ سرور:** زمانی که سرور شروع به ارسال اولین داده به مرورگر می‌کند.

مثال :

به این معنی است که از لحظه‌ای که مرورگر درخواست را به سرور ارسال می‌کند تا لحظه‌ای که اولین بایت : **TTFB = 200ms** داده دریافت می‌شود 200 میلی‌ثانیه طول کشیده است.

چرا مهم است؟

- زمان بالای TTFB می‌تواند نشان‌دهنده مشکلاتی مانند:
  - کندی سرور
  - مشکلات شبکه
  - زمان پردازش طولانی برای درخواست‌ها
- زمان پایین TTFB به بهبود تجربه کاربری و سرعت بارگذاری وب‌سایت کمک می‌کند.

چطور TTFB را بهبود دهیم؟

1. **استفاده از CDN** برای کاهش تأخیر شبکه
2. **بهینه‌سازی کدهای سرور و پایگاه داده**
3. **فعال کردن کشینگ (Caching)** در سمت سرور
4. **بهبود کانفیگ وب‌سرور (مثل NGINX یا Apache)**

FIREWALL

مفاهیم کلی : این مفاهیم متعلق به iptables یا فایروال لینوکس هستند و برای مدیریت ترافیک شبکه استفاده می‌شوند. به طور کلی، iptables ترافیک شبکه را بر اساس زنجیره‌ها (chains) و جداول (tables) مدیریت می‌کند. در ادامه، هر یک از موارد ذکرشده توضیح داده شده‌اند:

Chains (زنجیره‌ها):

1. PREROUTING

- این زنجیره قبل از اینکه ترافیک به شبکه داخلی یا پردازش‌های سیستم برسد، اعمال می‌شود.
- کاربرد: معمولاً برای تغییر مسیر (DNAT) یا تغییر بسته‌ها قبل از رسیدن به مقصد استفاده می‌شود.

2. INPUT

- ترافیکی که وارد سیستم یا سرور می‌شود و مقصد آن خود سرور است، از این زنجیره عبور می‌کند.
- کاربرد: برای کنترل دسترسی به سرویس‌ها یا پورت‌های سرور (مثلاً بستن یا باز کردن پورت‌ها).

3. OUTPUT

- ترافیکی که از سیستم خارج می‌شود و توسط خود سیستم تولید شده است، از این زنجیره عبور می‌کند.
- کاربرد: کنترل ترافیکی که از سرور به مقصدهای دیگر ارسال می‌شود.

4. POSTROUTING

- این زنجیره بعد از مسیریابی ترافیک و درست قبل از خروج از دستگاه اعمال می‌شود.
- کاربرد: معمولاً برای تغییر آدرس مبدأ (SNAT) استفاده می‌شود.

5. FORWARD

- ترافیکی که از سیستم عبور می‌کند ولی مقصد آن سیستم نیست، از این زنجیره عبور می‌کند.
- کاربرد: در سیستم‌های مسیریاب (router) برای انتقال ترافیک بین شبکه‌ها.

Tables (جداول):

1. Filter Table

- این جدول پیش‌فرض برای فیلتر کردن و مدیریت ترافیک استفاده می‌شود.
- زنجیره‌های مرتبط: `INPUT`, `OUTPUT`, `FORWARD`.
- کاربرد:
  - اجازه یا جلوگیری از دسترسی به ترافیک خاص.
  - بستن یا بازکردن پورت‌ها.

2. NAT Table

- این جدول برای ترجمه آدرس شبکه (Network Address Translation) استفاده می‌شود.
- زنجیره‌های مرتبط: `PREROUTING`, `POSTROUTING`, `OUTPUT`.
- کاربرد:

- **DNAT**: تغییر آدرس مقصد بسته‌ها (مثلاً برای Port Forwarding).
- **SNAT**: تغییر آدرس مبدأ بسته‌ها (مثلاً برای اینترنت به اشتراک گذاشته شده).

3. Mangle Table

- این جدول برای تغییر ویژگی‌های بسته‌های شبکه استفاده می‌شود.
- زنجیره‌های مرتبط: همه زنجیره‌ها (`PREROUTING`, `INPUT`, `FORWARD`, `OUTPUT`, `POSTROUTING`).
- کاربرد:
  - تغییر TTL (Time to Live).
  - علامت‌گذاری (marking) بسته‌ها برای پردازش‌های بعدی.

مثال کاربردی:

1. **PREROUTING**: اگر بخواهید ترافیک ورودی به یک آدرس IP مشخص را به آدرس دیگری بفرستید (DNAT)
2. **INPUT**: اگر بخواهید فقط به درخواست‌های SSH از یک محدوده IP اجازه دسترسی بدهید
3. **OUTPUT**: محدود کردن دسترسی سرور به اینترنت برای پروتکل یا مقصد خاص
4. **POSTROUTING**: استفاده از SNAT برای تغییر آدرس مبدأ بسته‌ها در اینترنت
5. **FORWARD**: تنظیم مسیریابی بین دو شبکه داخلی

خلاصه:

- **Chains** مشخص می‌کنند که ترافیک در کدام مرحله بررسی می‌شود.
- **Tables** (فیلتر، NAT، یا تغییرات پیشرفته) تعیین می‌کنند که چه کاری باید روی بسته انجام شود.

1. iptables

ابزاری قدیمی‌تر برای مدیریت فایروال در لینوکس است که مبتنی بر Netfilter کار می‌کند.

از iptables برای تعریف قوانینی (Rules) استفاده می‌شود که ترافیک شبکه را در سطح بسته (Packet) کنترل می‌کنند.

می‌توان قوانین را برای فیلتر کردن بسته‌ها، تغییر مسیر بسته‌ها، یا تنظیم NAT تعیین کرد.

نقاط قوت:

- بسیار قدرتمند و قابل تنظیم.
- امکان اعمال کنترل دقیق روی بسته‌های شبکه.

نقاط ضعف:

- پیچیدگی در مدیریت قوانین زیاد.
- مناسب نبودن برای محیط‌های پویا که تغییرات سریع نیاز دارند.

معماری:

- از **Chain**ها و **Table**ها برای کنترل بسته‌ها استفاده می‌کند.
- جدول‌ها شامل:

- **Filter Table**: برای فیلتر کردن بسته‌ها
- **NAT Table**: برای ترجمه آدرس شبکه
- **Mangle Table**: برای تغییر بسته‌های شبکه

Chain در iptables چیست؟

Chainها مانند لیست‌هایی از دستورات هستند که مشخص می‌کنند بسته‌های شبکه چگونه پردازش شوند. اگر بسته‌ای وارد یک Chain شود:

1. قوانین به ترتیب بررسی می‌شوند.
2. اولین قانونی که با بسته مطابقت داشته باشد اجرا می‌شود.
3. اگر هیچ قانونی مطابقت نداشته باشد، تصمیم نهایی به **Policy پیش‌فرض** Chain (مانند ACCEPT یا DROP) واگذار می‌شود.

Chain های پیش‌فرض در iptables:

1. **INPUT**: برای پردازش بسته‌های ورودی (Inbound Traffic)
2. **OUTPUT**: برای پردازش بسته‌های خروجی (Outbound Traffic)
3. **FORWARD**: برای بسته‌هایی که از سیستم عبور می‌کنند (مانند روتر)

زیر هم بودن قوانین در Chain:

زمانی که قوانین زیادی در یک Chain دارید:

1. **به ترتیب پردازش می‌شوند**:
   - هر بسته از ابتدای لیست شروع شده و تک‌تک قوانین را بررسی می‌کند.
   - اگر قانون شماره 50 با بسته مطابقت داشته باشد، بسته باید از 49 قانون قبلی عبور کند.
2. **تأثیر بر عملکرد**:
   - هرچه تعداد قوانین بیشتر باشد، زمان پردازش هر بسته افزایش می‌یابد.
   - این موضوع می‌تواند به تأخیر در پردازش بسته‌ها و کاهش کارایی شبکه منجر شود.
3. **قابلیت مدیریت پایین**:
   - با افزایش تعداد قوانین، مدیریت، عیب‌یابی، و تغییر قوانین دشوارتر می‌شود.
   - اشتباهات انسانی بیشتر رخ می‌دهد، مثلاً قرار دادن یک قانون در جای نادرست می‌تواند کل سیستم را مختل کند.

مشکلات زیاد بودن قوانین:

1. **کاهش عملکرد (Performance)**:
   - هر بسته باید تمام قوانین را یک‌به‌یک بررسی کند تا مطابقت پیدا کند.
   - این تأخیر در سیستم‌هایی با حجم بالای ترافیک (High Traffic) مشهودتر است.
2. **پیچیدگی در عیب‌یابی**:
   - پیدا کردن قانونی که یک مشکل خاص را ایجاد می‌کند دشوار است.
   - تغییر یک قانون ممکن است به قوانین دیگر آسیب برساند.
3. **خطاهای منطقی**:
   - اگر قوانین درست به ترتیب قرار نگیرند، ممکن است قوانین بعدی هرگز اجرا نشوند.
   - مثلاً اگر در ابتدای Chain قانونی نوشته شود که همه ترافیک را بپذیرد (ACCEPT)، قوانین بعدی بی‌اثر خواهند شد.
4. **مشکل در مقیاس‌پذیری**:
   - وقتی تعداد قوانین افزایش یابد، نگهداری و گسترش سیستم سخت‌تر می‌شود.
   - در محیط‌های بزرگ، مثل Data Centerها، این مشکل باعث محدودیت در طراحی شبکه می‌شود.

همچنین این امکان وجود دارد که تمام **rule** ها را به‌صورت یکجا در یک فایل ذخیره کرد و سپس این فایل را به iptables اعمال کرد. این روش به شما اجازه می‌دهد قوانین را بهتر مدیریت و تغییر دهید. همچنین باعث می‌شود که نیازی نباشد هر قانون را به‌صورت جداگانه اجرا کنید. این کار به‌خصوص در محیط‌های بزرگ و پیچیده بسیار مفید است.

ذخیره تمام Ruleها در یک فایل

برای ذخیره تمام قوانین فعلی iptables در یک فایل، می‌توانید از دستور زیر استفاده کنید:

```
iptables-save > /etc/iptables/rules.v4
```

بارگذاری قوانین از فایل

برای اعمال مجدد قوانین ذخیره‌شده، از دستور زیر استفاده کنید:

```
iptables-restore < /etc/iptables/rules.v4
```

2. firewalld

1. **ابزاری مدرن‌تر** برای مدیریت فایروال است که روی iptables ساخته شده و کار با فایروال را ساده‌تر می‌کند.
2. **دایمون (Daemon)** که توسط firewalld استفاده می‌شود، امکان مدیریت قوانین فایروال را به‌صورت پویا و بدون نیاز به ری‌استارت فراهم می‌کند.

مزایا

1. **رابط کاربری ساده‌تر** نسبت به iptables ارائه می‌دهد.
2. **ناحیه‌های فایروال (Zones)** را برای دسته‌بندی قوانین بر اساس نوع شبکه (مانند Work، Public، Home و...) فراهم می‌کند.
3. **امکان تغییر قوانین** به‌صورت پویا وجود دارد، بدون اینکه اتصال شبکه مختل شود.

معایب

1. **برای محیط‌های پیچیده** یا بسیار حساس، ممکن است ویژگی‌های محدودی داشته باشد.

ویژگی‌ها

1. **به‌صورت پیش‌فرض** در توزیع‌های مدرن لینوکس مانند RHEL و Fedora استفاده می‌شود.
2. **سازگاری با iptables** را نیز ارائه می‌دهد.

3. nftables

3. **توضیحات:**

nftables به‌عنوان جایگزین مدرن برای iptables و ip6tables معرفی شده و به‌طور مستقیم توسط Netfilter پشتیبانی می‌شود. هدف اصلی nftables، ساده‌سازی قوانین شبکه و افزایش کارایی است.

مزایا

1. **استفاده از یک هسته واحد** برای مدیریت ipv4 و ipv6.
2. **کاهش پیچیدگی قوانین:** در مقایسه با iptables، قوانین ساده‌تر و واضح‌تر نوشته می‌شوند.
3. **عملکرد بالا:** نیاز به بررسی مکرر جدول‌ها کاهش یافته و سرعت عملکرد بهبود می‌یابد.
4. **پشتیبانی از حالت‌های پویا** و انعطاف‌پذیری بیشتر.

معایب

1. **هنوز به اندازه iptables** در برخی محیط‌های سنتی محبوب نیست.
2. **یادگیری آن زمان‌بر** ممکن است برای افرادی که به iptables عادت دارند، چالش‌برانگیز باشد.

ویژگی‌ها

1. **در سیستم‌های جدید لینوکسی** (مانند Ubuntu 20.04 یا RHEL 8) به‌عنوان جایگزین پیش‌فرض iptables استفاده می‌شود.
2. **امکان تعریف قوانین** با Syntax ساده‌تر و خوانا تر فراهم شده است.

"Configuring Firewall Rules with iptables"

```
# Flush previous rules
iptables -F

# Set default policies to DROP (default action for inbound, outbound, and
forwarded packets)

iptables -P INPUT DROP / ACCEPT
iptables -P OUTPUT DROP / ACCEPT
iptables -P FORWARD DROP / ACCEPT

# Allow incoming SSH on port 22
iptables -A INPUT -p tcp --dport 22 -j ACCEPT

# Allow incoming HTTP traffic on port 80
iptables -A INPUT -p tcp --dport 80 -j ACCEPT

# Allow incoming HTTPS traffic on port 443
iptables -A INPUT -p tcp --dport 443 -j ACCEPT

# Allow incoming ping (ICMP) requests
```

```
iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEPT

# Drop all incoming traffic from a specific IP address (e.g., 192.168.1.100)
iptables -A INPUT -s 192.168.1.100 -j DROP
```

- `-A` **(Append)**: اضافه کردن قانون به انتهای زنجیره.
  - مثال: `iptables -A INPUT -p tcp --dport 80 -j ACCEPT`
- `-I` **(Insert)**: وارد کردن قانون در ابتدای زنجیره (یا در موقعیت مشخص).
  - مثال: `iptables -I INPUT 1 -p tcp --dport 80 -j ACCEPT`
  - `1` مشخص‌کننده موقعیت در زنجیره است.
- `-D` **(Delete)**: حذف یک قانون از زنجیره.
  - مثال: `iptables -D INPUT -p tcp --dport 80 -j ACCEPT`
- `-F` **(Flush)**: پاک کردن تمام قوانین از زنجیره.
  - مثال: `iptables -F`
- `-P` **(Policy)**: تنظیم سیاست پیش‌فرض برای زنجیره.
  - مثال: `iptables -P INPUT DROP`
- `-L` **(List)**: نمایش قوانین موجود در زنجیره.
  - مثال: `iptables -L`
- `-T` **(Table)**: مشخص کردن جدول (برای مثال `filter` یا `nat`).
  - مثال: `iptables -t nat -L`
- `-N` **(New Chain)**: ایجاد یک زنجیره جدید.
  - مثال: `iptables -N MYCHAIN`
- `-X` **(Delete Chain)**: حذف یک زنجیره تعریف‌شده توسط کاربر.
  - مثال: `iptables -X MYCHAIN`
- `-E` **(Rename Chain)**: تغییر نام یک زنجیره.
  - مثال: `iptables -E MYCHAIN NEWCHAIN`
- `-s` **(Source)**: تعیین آدرس IP مبدا برای قانون.
  - مثال: `iptables -A INPUT -s 192.168.1.100 -j DROP`
- `-d` **(Destination)**: تعیین آدرس IP مقصد برای قانون.
  - مثال: `iptables -A OUTPUT -d 192.168.1.200 -j ACCEPT`
- `-p` **(Protocol)**: تعیین پروتکل (TCP، UDP، ICMP و غیره).
  - مثال: `iptables -A INPUT -p tcp --dport 22 -j ACCEPT`
- `--dport` / `--sport` **(Destination Port / Source Port)**: تعیین پورت مقصد یا مبدا.
  - مثال: `iptables -A INPUT -p tcp --dport 80 -j ACCEPT`
- `-j` **(Jump)**: تعریف اکشن برای بسته‌ها (مانند ACCEPT، DROP، REJECT).
  - مثال: `iptables -A INPUT -p tcp --dport 80 -j ACCEPT`

- **`-m` (Match)**: استفاده از ماژول‌های اضافی برای تطابق دقیق‌تر.
  - مثال: `iptables -A INPUT -m state --state NEW -j ACCEPT`
- **`--state`**: تنظیم وضعیت اتصال (مثلاً NEW، ESTABLISHED).
  - مثال: `iptables -A INPUT -m state --state NEW -j ACCEPT`
- **`-v` (Verbose)**: نمایش اطلاعات بیشتر در مورد قوانین.
  - مثال: `iptables -L -v`
- **`-h` (Help)**: نمایش صفحه راهنما.
  - مثال: `iptables -h`
- `INPUT DROP`: تمام ترافیک ورودی به‌طور پیش‌فرض مسدود می‌شود
- `INPUT ACCEPT`: تمام ترافیک ورودی به‌طور پیش‌فرض مجاز می‌شود
- `OUTPUT ACCEPT`: تمام ترافیک خروجی به‌طور پیش‌فرض مجاز است
- `OUTPUT DROP`: تمام ترافیک خروجی به‌طور پیش‌فرض مسدود می شود
- `FORWARD DROP`: تمام ترافیک عبوری از سیستم مسدود می‌شود

```
iptables -F
```

این دستور تمام قوانین قبلی فایروال را از جدول `iptables` پاک می‌کند.

```
iptables -P INPUT DROP / ACCEPT
iptables -P OUTPUT DROP / ACCEPT
iptables -P FORWARD DROP / ACCEPT
```

این دستورات سیاست‌های پیش‌فرض فایروال را تنظیم می‌کنند:

```
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
```

این دستور به ترافیک ورودی با پروتکل TCP روی پورت 22 (SSH) اجازه عبور می‌دهد. پورت 22 برای اتصال به سرور از طریق پروتکل SSH استفاده می‌شود.

```
iptables -A INPUT -p tcp --dport 80 -j DROP
```

این دستور ترافیک ورودی برای پروتکل TCP روی پورت 80 (HTTP) را مسدود می کند . این پورت برای دسترسی به وب‌سایت‌ها از طریق HTTP استفاده می‌شود.

```
iptables -A INPUT -p tcp --dport 443 -j ACCEPT
```

این دستور به ترافیک ورودی برای پروتکل TCP روی پورت 443 (HTTPS) اجازه می‌دهد. پورت 443 برای اتصال امن به وب‌سایت‌ها استفاده می‌شود.

```
iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEPT
```

این دستور اجازه ارسال درخواست‌های پینگ را می دهد .

```
iptables -A OUTPUT -p icmp --icmp-type echo-reply -j DROP
```

این دستور تمام درخواست‌های **پاسخ پینگ (Echo Reply)** که به سرور شما می‌آیند را مسدود کند .

```
iptables -A INPUT -s 192.168.1.100 -j DROP
```

این دستور ترافیک ورودی از یک آدرس IP خاص (در اینجا 192.168.1.100) را مسدود می‌کند.

در فایروال‌ها و به‌ویژه در **iptables**، دستورات **REJECT** و **DROP** هر دو برای مسدود کردن ترافیک استفاده می‌شوند، اما تفاوت‌های مهمی در نحوه رفتار آن‌ها وجود دارد. این تفاوت‌ها بر نحوه مدیریت و پاسخ‌دهی به بسته‌ها تأثیر می‌گذارد.

1. DROP:

- زمانی که یک بسته با دستور **DROP** مسدود می‌شود، هیچ پاسخی به فرستنده ارسال نمی‌شود.
- بسته به‌سادگی از بین می‌رود و هیچگونه اطلاع‌رسانی به فرستنده مبنی بر مسدود شدن بسته وجود ندارد.
- این رفتار باعث می‌شود که فرستنده نمی‌تواند متوجه شود که بسته‌ای که ارسال کرده به‌طور عمدی مسدود شده یا به‌دلیل سایر مشکلات (مانند خطا در مسیر) از دست رفته است.

مزایا:

- باعث کاهش بار سرور یا شبکه می‌شود، زیرا هیچ اطلاعات اضافی (برای مسدود کردن بسته) به فرستنده ارسال نمی‌شود.
- این معمولاً برای امنیت بیشتر به‌کار می‌رود، زیرا مهاجم ممکن است نتواند تشخیص دهد که بسته‌ها عمداً مسدود می‌شوند یا به‌دلیل مشکلات دیگر از دست رفته‌اند.

2. REJECT:

- زمانی که یک بسته با دستور **REJECT** مسدود می‌شود، سیستم یک پاسخ به فرستنده ارسال می‌کند که به آن اطلاع می‌دهد که بسته مسدود شده است.
- بسته‌ها به‌طور مستقیم رد می‌شوند و یک پیام خطا (مثلاً "ICMP "Destination Unreachable یا TCP ""Connection Refused) به فرستنده ارسال می‌شود.
- این پیام به فرستنده اطلاع می‌دهد که مقصد دسترس‌پذیر نیست یا اتصال رد شده است.

مزایا:

- این رفتار برای جلوگیری از سردرگمی فرستندگان مفید است، زیرا فرستنده می‌تواند متوجه شود که بسته به‌طور عمدی مسدود شده است.
- در برخی موارد، این می‌تواند برای شبکه‌های مدیریتی مفید باشد تا مطمئن شوند که دیگر کاربران یا سیستم‌ها خطای مسیریابی دریافت نمی‌کنند و علت مسدود شدن بسته‌ها مشخص است.

ICMP (ECHO)

در پروتکل ICMP (Internet Control Message Protocol)، پیام‌های **Echo** و **Echo Reply** برای تست اتصال شبکه و بررسی تاخیر استفاده می‌شوند. این پیام‌ها معمولاً در دستور `ping` مشاهده می‌شوند.

ساختار یک پیام ICMP :

نوع (Type): مشخص‌کننده نوع پیام (مثلاً Echo Reply، Echo Request و غیره).
کد (Code): اطلاعات اضافی در مورد نوع پیام را ارائه می‌دهد.
مجموع بررسی (Checksum): برای اطمینان از صحت داده‌ها استفاده می‌شود.
بقیه هدر (Rest of Header): بسته به نوع و کد متغیر است (مثلاً شامل شناسه و ترتیب برای پیام‌های اکو).

انواع پیام‌های Echo در ICMP:

1. **Echo Request** (نوع 8):
   - زمانی که شما از دستور `ping` برای ارسال پینگ به یک دستگاه دیگر استفاده می‌کنید، در واقع شما یک **Echo Request** ارسال کرده‌اید.
   - این پیام به دستگاه مقصد ارسال می‌شود تا از آن خواسته شود که آیا به شبکه متصل است یا خیر.
   - این پیام به صورت یک درخواست به سیستم مقصد ارسال می‌شود تا نشان دهد که سیستم ارسال‌کننده (مبدا) منتظر دریافت پاسخ است.
2. **Echo Reply** (نوع 0):
   - زمانی که دستگاه مقصد درخواست پینگ را دریافت می‌کند، به آن پاسخ می‌دهد. پاسخ دریافتی از دستگاه مقصد **Echo Reply** است.
   - این پاسخ نشان می‌دهد که دستگاه مقصد آماده است و به شبکه متصل است.
   - در واقع، پاسخ به یک **Echo Request** که به دستگاه مقصد ارسال شده، پیام **Echo Reply** است.

روند عملکرد:

- زمانی که شما از دستور `ping` استفاده می‌کنید، یک **Echo Request** به سرور هدف ارسال می‌شود.
- اگر سرور هدف در دسترس باشد، یک **Echo Reply** به مبدا ارسال می‌شود.
- تاخیر زمانی بین ارسال **Echo Request** و دریافت **Echo Reply** به شما اطلاعاتی در مورد زمان تاخیر و وضعیت اتصال شبکه می‌دهد.

مثال:

- **Echo Request**: `ping example.com` یا `ping 192.168.1.1`

- **Echo Reply**: شما در پاسخ به این دستور پیامی مانند این دریافت خواهید کرد:

```
bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=0.040 ms
```

سایر انواع ICMP:

- **Destination Unreachable**: زمانی که دستگاه مقصد نتواند به هدف برسد
- **Time Exceeded**: زمانی که بسته داده به مقصد نمی‌رسد و در راه از بین می‌رود
- **Redirect**: میزبان را هدایت می‌کند تا از یک روتر دیگر برای بهبود مسیریابی استفاده کند

NameSpaces

Namespaces در لینوکس تکنولوژی‌ای هستند که برای ایزوله کردن منابع سیستم استفاده می‌شوند. هر namespace به یک نوع خاص از منابع مربوط می‌شود و به فرآیندها اجازه می‌دهد تا تصور کنند در یک محیط جداگانه و مخصوص به خودشان هستند. در ادامه، توضیح ساده‌ای از انواع namespaces ارائه می‌شود:

1. mnt (Mount Namespace)

- **کاربرد:** ایزوله کردن **سیستم فایل**.
- **توضیح ساده:** هر فرآیند می‌تواند سیستم فایل مخصوص خودش را داشته باشد. مثلاً فرآیند A می‌تواند یک درایو خاص را mount کند و فرآیند B آن را نبیند.
- **مثال:** در Docker، کانتینرها سیستم فایل جداگانه‌ای دارند و تغییرات در آن‌ها روی سیستم اصلی تأثیری ندارد.

2. pid (Process ID Namespace)

- **کاربرد:** ایزوله کردن **شناسه فرآیندها (PIDs)**.
- **توضیح ساده:** هر namespace می‌تواند مجموعه‌ای از فرآیندها با PIDs مختص به خودش داشته باشد. فرآیندها در namespaceهای مختلف نمی‌توانند فرآیندهای یکدیگر را ببینند.
- **مثال:** در کانتینرها، فرآیند "1" مربوط به همان کانتینر است و فرآیندهای سیستم اصلی را نمی‌بیند.

3. net (Network Namespace)

- **کاربرد:** ایزوله کردن **تنظیمات شبکه**.
- **توضیح ساده:** هر namespace شبکه خودش را دارد، شامل **آدرس‌های IP، routing tables، و پورت‌ها**.
- **مثال:** در Docker، هر کانتینر می‌تواند آدرس IP خودش را داشته باشد، بدون اینکه با شبکه سیستم اصلی تداخل پیدا کند.

4. user (User Namespace)

- **کاربرد:** ایزوله کردن **شناسه‌های کاربری (UID و GID)**.
- **توضیح ساده:** یک فرآیند می‌تواند در namespace خودش کاربر root باشد، اما در سیستم اصلی همچنان به‌عنوان یک کاربر معمولی دیده شود.

- **مثال:** کانتینرها برای امنیت بیشتر از این ویژگی استفاده می‌کنند تا فرآیندهای داخل کانتینر دسترسی مستقیم به سیستم اصلی نداشته باشند.

5. Time Namespace (یا UTS Namespace)

کاربرد: ایزوله کردن زمان سیستم.

توضیح ساده: هر **namespace** می‌تواند زمان خودش را تنظیم کند، بدون اینکه زمان سیستم اصلی تغییر کند. این ویژگی به فرآیندها این امکان را می‌دهد که برای تست یا شبیه‌سازی، تصور کنند که در یک منطقه زمانی متفاوت یا ساعت دیگری از روز قرار دارند.

مثال: یک فرآیند در داخل یک کانتینر می‌تواند تصور کند که در ساعت دیگری از روز است، بدون اینکه این تغییرات زمان بر سیستم اصلی تأثیر بگذارد.

DOCKER