

بسم الله الرحمن الرحيم

فاطمه اثباتی

هدف پروژه

هدف از این پروژه، ایجاد یک linux shell script است که فرایند log rotation را در محیط لینوکس شبیه سازی کند. این اسکریپت باید فایلی را که در اختیار تان قرار میگیرد، از انتها به صورت خط به خط بخواند و بر اساس معیارهای ذکر شده در فایل های جداگانه سازماندهی کند.

linux shell script چیست؟

Linux Shell Script یک برنامه یا فایل متنی است که حاوی مجموعه‌ای از دستورات است که در محیط خط فرمان (Shell) سیستم عامل لینوکس یا دیگر سیستم‌های یونیکس-مانند اجرا می‌شود. این اسکریپت‌ها به برنامه‌نویسان و کاربران اجازه می‌دهند تا وظایف مختلف را به صورت خودکار انجام دهند. به عبارت دیگر، Shell Scripting راهی برای نوشتن و اجرای دستوراتی است که به صورت متوالی و بدون نیاز به مداخله انسانی انجام می‌شوند.

Shell چیست؟

Shell در سیستم‌عامل‌های لینوکس یک واسط کاربری است که بین کاربر و هسته (Kernel) سیستم‌عامل قرار دارد. این واسط به کاربر اجازه می‌دهد دستورات را وارد کرده و با سیستم عامل ارتباط برقرار کند. مشهورترین شل‌ها عبارتند از:

- Bash (Bourne Again Shell) محبوب‌ترین شل در لینوکس.
- Zsh یک شل پیشرفته که امکانات بیشتری نسبت به Bash ارائه می‌دهد.
- Ksh (Korn Shell) و Csh (C Shell) نیز از دیگر شل‌های پرکاربرد هستند.

قسمت اول:

- فایل `file.log` باید از انتها به صورت خط به خط خوانده و پردازش شود.
- فایل `file.log` باید بر اساس نوع لاگ (`INFO` , `WARNING` , `ERROR`) در فایل‌های جداگانه دسته‌بندی شود.
- برای مثال نحوه نام‌گذاری فایل‌ها به صورت زیر باشد:

<code>log-error.log</code>	<code>log-warning.log</code>	<code>log-info.log</code>
<code>log-error-1.log</code>	<code>log-warning-1.log</code>	<code>log-info-1.log</code>
<code>log-error-2.log</code>	<code>log-warning-2.log</code>	<code>log-info-2.log</code>
<code>log-error-3.log</code>		<code>log-info-3.log</code>
<code>log-error-4.log</code>		

- نحوه قرارگیری لاگ‌ها در فایل‌ها به این صورت باشد که فایل `log-error.log` شامل جدیدترین لاگ‌ها و فایل `log-error-n.log` (برای مثال `log-error-2.log`) شامل قدیمی‌ترین لاگ‌ها باشد.
- حجم هیچ فایل‌ی نباید از ۱ مگابایت بیشتر شود.
- زمانی که حجم فایل به حد نصاب مشخص شده رسید، یک فایل جدید ایجاد کنید و محتوای فایل را به فایل جدید منتقل کنید (بر اساس نام‌گذاری تعیین‌شده، فایل‌های جدید را نام‌گذاری کنید) تا فایل جاری (مثلاً `log-error.log`) خالی شود و آماده دریافت لاگ‌های جدید شود.

```
#!/bin/bash

# Define the main log file and the categorized log files
LOG_FILE="file.log"
INFO_LOG="log-info.log"
WARNING_LOG="log-warning.log"
ERROR_LOG="log-error.log"

# Define the maximum size for each log file (1 MB in bytes)
MAX_SIZE=1048576

# Function to rotate log files if they exceed the maximum size
rotate_log() {
    local log_file=$1
    local base_name=$(basename "$log_file" .log) # Get the base name of the log file
    local count=1
```

```

# Find the next available suffix for the rotated log file
while [ -f "${base_name}-${count}.log" ]; do
    ((count++))
done

# Move the current log file to a new file with the incremental suffix
mv "$log_file" "${base_name}-${count}.log"
touch "$log_file" # Create a new empty log file
}

# Function to process the main log file and categorize logs
process_log() {
    local log_type=$1
    local log_file=$2

    # Read the main log file from the end using tac and filter lines based on the
log type
    while IFS= read -r line; do
        echo "$line" >> "$log_file" # Append the line to the corresponding log
file
        if [ $(stat -c%s "$log_file") -gt $MAX_SIZE ]; then # Check if the log
file exceeds the maximum size
            rotate_log "$log_file" # Rotate the log file if it exceeds the
maximum size
        fi
    done < <(tac "$LOG_FILE" | grep "$log_type") # Filter lines based on the log
type
}

# Process INFO logs
process_log "INFO" "$INFO_LOG"

# Process WARNING logs
process_log "WARNING" "$WARNING_LOG"

# Process ERROR logs
process_log "ERROR" "$ERROR_LOG"

```

حالا جزء به جزء اسکریپتی که نوشته شده را بررسی میکنیم:

(۱) تعریف فایل‌های لاگ اصلی و لاگ‌های دسته‌بندی‌شده:

```
LOG_FILE="file.log"
INFO_LOG="log-info.log"
WARNING_LOG="log-warning.log"
ERROR_LOG="log-error.log"
```

LOG_FILE: نام فایل اصلی لاگ که همه‌ی پیام‌ها در آن ثبت شده‌اند (در این مثال file.log).
INFO_LOG: نام فایلی که لاگ‌های مربوط به نوع INFO در آن ذخیره می‌شوند.
WARNING_LOG: نام فایلی که لاگ‌های مربوط به نوع WARNING در آن ذخیره می‌شوند.
ERROR_LOG: نام فایلی که لاگ‌های مربوط به نوع ERROR در آن ذخیره می‌شوند.

(۲) تعریف حداکثر اندازه برای هر فایل لاگ:

```
MAX_SIZE=1048576
```

MAX_SIZE: مقدار حداکثر اندازه فایل‌های لاگ که در اینجا برابر با ۱ مگابایت (۱۰۴۸۵۷۶ بایت) تعیین شده است. اگر اندازه فایل از این مقدار بیشتر شود، لاگ جدید ایجاد می‌شود و لاگ قبلی با استفاده از تابع rotate_log چرخانده (rotated) می‌شود.

(۳) تعریف تابع چرخش (Rotate) فایل‌های لاگ:

```
rotate_log() {
    local log_file=$1
    local base_name=$(basename "$log_file" .log)
    local count=1
```

```

while [ -f "${base_name}-${count}.log" ]; do
    ((count++))
done

mv "$log_file" "${base_name}-${count}.log"
touch "$log_file"
}

```

rotate_log: این تابع برای چرخش فایل‌های لاگ استفاده می‌شود و در صورتی که اندازه فایل لاگ از مقدار حداکثر تعریف شده بیشتر شود، یک فایل جدید با شماره‌گذاری ایجاد می‌کند.

- **local log_file=\$1**: این خط، نام فایل لاگ را از ورودی تابع دریافت کرده و آن را به متغیر `log_file` اختصاص می‌دهد.
- **basename "\$log_file".log**: این دستور از فایل لاگ، نام فایل بدون پسوند `.log` را برمی‌گرداند. به عنوان مثال، اگر ورودی `log-info.log` باشد، خروجی `log-info` خواهد بود.
- **count=1**: مقدار اولیه برای شمارش فایل‌های چرخانده شده است.
- **while [-f "\${base_name}-\${count}.log"]**: این حلقه بررسی می‌کند که آیا فایل چرخانده شده‌ای با این شمارش وجود دارد یا خیر. اگر وجود داشت، شمارنده را افزایش می‌دهد تا شماره‌ی جدیدی پیدا کند که موجود نباشد.
- **mv "\$log_file" "\${base_name}-\${count}.log"**: فایل لاگ موجود را با استفاده از دستور `mv` به یک فایل جدید با شماره‌گذاری جدید منتقل می‌کند.
- **touch "\$log_file"**: این دستور یک فایل لاگ خالی با همان نام اولیه ایجاد می‌کند تا بتوان از آن برای ذخیره لاگ‌های جدید استفاده کرد.

(۴) تعریف تابع برای پردازش و دسته‌بندی لاگ‌ها:

```
process_log() {
    local log_type=$1
    local log_file=$2

    while IFS= read -r line; do
        echo "$line" >> "$log_file"
        if [ $(stat -c%s "$log_file") -gt $MAX_SIZE ]; then
            rotate_log "$log_file"
        fi
    done < <(tac "$LOG_FILE" | grep "$log_type")
}
```

process_log: این تابع وظیفه دارد که لاگ‌ها را از فایل اصلی بخواند و بر اساس نوع لاگ (ERROR، WARNING، INFO) آن‌ها را در فایل‌های جداگانه ذخیره کند. در عین حال، اگر اندازه هر فایل از حداکثر اندازه تعیین شده بیشتر شد، آن فایل را می‌چرخاند.

- **local log_type=\$1**: نوع لاگ (مانند INFO، WARNING، ERROR) را از ورودی تابع دریافت می‌کند.
- **local log_file=\$2**: نام فایل لاگی که باید لاگ‌های مربوط به نوع خاص را در آن ذخیره کند، از ورودی تابع دریافت می‌شود.
- **tac "\$LOG_FILE" | grep "\$log_type"**: این دستور ابتدا فایل لاگ اصلی را از انتها می‌خواند (با استفاده از دستور tac که برعکس cat است) و سپس خطوطی که با نوع لاگ مشخص شده مطابقت دارند (با استفاده از grep) فیلتر می‌کند. دلیل استفاده از tac، این است که اسکرپت خطوط جدیدتر را ابتدا پردازش کند.
- **IFS= read -r line**: دستور read به طور خط به خط فایل لاگ فیلترشده را می‌خواند. IFS= به این معنی است که فاصله‌ها و تب‌ها به عنوان جداکننده‌های خطوط در نظر گرفته نمی‌شوند. -r هم تضمین می‌کند که بک‌اسلش‌ها (\) نادیده گرفته نمی‌شوند.
- **echo "\$line" >> "\$log_file"**: خط فیلتر شده را به فایل لاگ مربوطه اضافه می‌کند.

- **stat -c%s "\$log_file"**: این دستور اندازه فایل لاگ را بر حسب بایت برمی گرداند. اگر اندازه فایل از مقدار MAX_SIZE بیشتر شد، فایل چرخانده می شود.

(۵) پردازش لاگ ها برای هر نوع لاگ:

```
process_log "INFO" "$INFO_LOG"  
process_log "WARNING" "$WARNING_LOG"  
process_log "ERROR" "$ERROR_LOG"
```




















این بخش سه بار تابع `process_log` را فراخوانی می کند تا لاگ های مربوط به هر نوع (INFO، WARNING، ERROR) را پردازش کند.

- **process_log "INFO" "\$INFO_LOG"**: لاگ های نوع INFO را از فایل اصلی می گیرد و در فایل `log-info.log` ذخیره می کند.
- **process_log "WARNING" "\$WARNING_LOG"**: لاگ های نوع WARNING را از فایل اصلی می گیرد و در فایل `log-warning.log` ذخیره می کند.
- **process_log "ERROR" "\$ERROR_LOG"**: لاگ های نوع ERROR را از فایل اصلی می گیرد و در فایل `log-error.log` ذخیره می کند.

```

esbati@esbati-ASUSLaptop:~/mci-hadoop$ time ./log_rotation.sh
real    5m47.059s
user    1m53.569s
sys     4m1.602s
esbati@esbati-ASUSLaptop:~/mci-hadoop$ 

```

Name	Size	Modified	
 DWBI_Department_Task.pdf	134.1 kB	13 ژانویه	☆
 file.log	15.0 MB	13 ژانویه	☆
 log_rotation.sh	1.6 kB	04:09	☆
 log-info-1.log	1.0 MB	10:11	☆
 log-info-2.log	1.0 MB	10:11	☆
 log-info-3.log	1.0 MB	10:12	☆
 log-info-4.log	1.0 MB	10:12	☆
 log-info.log	738.6 kB	10:12	☆
 log-warning-1.log	1.0 MB	10:13	☆
 log-warning-2.log	1.0 MB	10:13	☆
 log-warning-3.log	1.0 MB	10:14	☆
 log-warning-4.log	1.0 MB	10:14	☆
 log-warning-5.log	1.0 MB	10:14	☆
 log-warning.log	241.1 kB	10:14	☆
 log-error-1.log	1.0 MB	10:15	☆
 log-error-2.log	1.0 MB	10:15	☆
 log-error-3.log	1.0 MB	10:16	☆
 log-error-4.log	1.0 MB	10:16	☆
 log-error.log	387.4 kB	10:16	☆

Hadoop چیست؟

Hadoop یک فریم‌ورک متن‌باز است که برای ذخیره‌سازی و پردازش حجم عظیمی از داده‌ها به کار می‌رود. این فریم‌ورک توسط Apache Software Foundation توسعه داده شده و طراحی شده تا داده‌ها را به صورت توزیع‌شده و موازی بر روی چندین سرور یا سیستم پردازش کند.

Hadoop به‌طور کلی از چهار بخش اصلی تشکیل شده است:

۱. Hadoop Distributed File System (HDFS): یک سیستم فایل توزیع‌شده که داده‌ها را به بلوک‌های کوچکی تقسیم می‌کند و آن‌ها را در سرورهای مختلف (nodes) ذخیره می‌کند. این سیستم از قابلیت تحمل خطا (اشکال‌پذیری) برخوردار است، به این معنا که در صورت از دست رفتن یک نود، نسخه‌های دیگری از داده وجود دارد.

۲. MapReduce: یک مدل برنامه‌نویسی برای پردازش موازی داده‌ها در سرورهای مختلف. MapReduce به دو بخش تقسیم می‌شود:

- Map: داده‌ها را به قسمت‌های کوچکی تقسیم و پردازش می‌کند.
- Reduce: نتایج پردازش شده را جمع‌بندی و تجمیع می‌کند.

۳. YARN (Yet Another Resource Negotiator): بخشی از Hadoop که مدیریت منابع و برنامه‌های کاربردی را در سیستم‌های توزیع‌شده انجام می‌دهد.

۴. Hadoop Common: مجموعه‌ای از ابزارها و کتابخانه‌های عمومی که از دیگر بخش‌های Hadoop پشتیبانی می‌کنند.

Hadoop به دلیل مقیاس‌پذیری بالا و توانایی مدیریت داده‌های بسیار بزرگ (Big Data)، به‌ویژه در حوزه‌هایی مثل تحلیل داده‌ها، داده‌کاوی و یادگیری ماشین، به‌طور گسترده مورد استفاده قرار می‌گیرد.

قسمت دوم:

- Hadoop نسخه 3.3.4 از این [لینک](#) دریافت و نصب کنید.
- یک دایرکتوری با نام log در hdfs بسازید
- تمام فایل‌هایی که در مرحله قبل ایجاد شده را در دایرکتوری ایجاد شده در hdfs قرار دهید.
- دستور `hdfs dfs -ls /log` را در سیستم خود اجرا کنید. فایل‌های مورد نظر باید در این دایرکتوری قابل مشاهده باشد. اسکرین شات خروجی این دستور را به مستندات مرحله دو اضافه کنید.

مراحل نصب Hadoop:













ابتدا از نصب بودن جاوا روی سیستم خود اطمینان حاصل می‌فرماییم:

```
esbati@esbati-ASUSLaptop:~$ java -version
openjdk version "1.8.0_422"
OpenJDK Runtime Environment (build 1.8.0_422-8u422-b05-1~22.04-b05)
OpenJDK 64-Bit Server VM (build 25.422-b05, mixed mode)
esbati@esbati-ASUSLaptop:~$
```

و در PATH سیستم نیز آدرس آن را اضافه می‌کنیم:

```
D export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
D export PATH=$PATH:$JAVA_HOME/bin
```

حال که سیستم آماده است از سایت اصلی آپاچی Hadoop را دانلود و از حالت فشرده خارج می‌کنیم:

https://dlcdn.apache.org/hadoop/common/			
Index of /hadoop/common			
Name	Last modified	Size	Description
 Parent Directory		-	
 current/	2024-03-17 06:51	-	
 current2/	2022-06-17 11:29	-	
 hadoop-2.10.2/	2022-06-17 11:29	-	
 hadoop-3.2.4/	2022-07-22 02:08	-	
 hadoop-3.3.5/	2023-03-24 10:56	-	
 hadoop-3.3.6/	2023-06-26 00:46	-	
 hadoop-3.4.0/	2024-03-17 06:51	-	
 stable/	2024-03-17 06:51	-	
 stable2/	2022-06-17 11:29	-	
 KEYS	2024-01-04 04:04	426K	
 readme.txt	2015-04-21 01:32	184	

حال نیاز است متغیر های محیطی (محیط سیستم) را برای Hadoop در `bashrc` تعریف کنیم و سپس تغییرات را اعمال کنیم، با استفاده از دستورات زیر:

```
nano ~/.bashrc
source ~/.bashrc
```

```
export HADOOP_HOME=/usr/local/hadoop
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export LD_LIBRARY_PATH=$HADOOP_COMMON_LIB_NATIVE_DIR:$LD_LIBRARY_PATH
```

حال با اعمال تغییرات که تنظیمات پیش از اجرا هستند و در سایت اصلی به آن اشاره شده است، تنظیمات فرموده شده را به چهار فایل زیر اضافه میکنیم:

```
nano $HADOOP_HOME/etc/hadoop/core-site.xml
nano $HADOOP_HOME/etc/hadoop/hdfs-site.xml
nano $HADOOP_HOME/etc/hadoop/mapred-site.xml
nano $HADOOP_HOME/etc/hadoop/yarn-site.xml
```

core-site.xml: این فایل تنظیمات اصلی و پایه سیستم Hadoop را مشخص می کند. به عنوان مثال، این فایل شامل تنظیمات مربوط به سیستم فایل توزیع شده (HDFS) و نحوه دسترسی به آن است.

hdfs-site.xml: این فایل پیکربندی سیستم فایل توزیع شده (HDFS) را تعیین می کند. HDFS یک سیستم فایل توزیع شده است که Hadoop برای ذخیره سازی داده های بزرگ استفاده می کند.

mapred-site.xml: این فایل مربوط به تنظیمات MapReduce، چارچوبی برای پردازش موازی داده ها در Hadoop است. در این فایل، پارامترهای مربوط به MapReduce تنظیم می شوند.

yarn-site.xml: این فایل برای تنظیمات YARN (Yet Another Resource Negotiator)، سیستم مدیریت منابع در Hadoop، استفاده می شود. YARN مسئول مدیریت منابع در خوشه Hadoop و اجرای وظایف پردازشی در آن است.

مرحله بعد: راه اندازی SSH بدون رمز عبور (Passwordless SSH)

راه اندازی SSH بدون رمز عبور (Passwordless SSH) در بسیاری از سناریوها، به ویژه در محیط های توزیع شده مانند **Hadoop** یا دیگر سیستم های توزیع شده، به کار می رود.

```
mci-hadoop@esbati-ASUSLaptop:~$ ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
Generating public/private rsa key pair.
Created directory '/home/mci-hadoop/.ssh'.
Your identification has been saved in /home/mci-hadoop/.ssh/id_rsa
Your public key has been saved in /home/mci-hadoop/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:cmhVfU3nwowHYLtp26z69Dqq0sRg9k0p/DJQj+gN2U mci-hadoop@esbati-ASUSLaptop
The key's randomart image is:
+---[RSA 3072]-----+
|  + E . +0. oo|
| o & = o ..=.o|
| . * @ . . . = .|
| . o * o . .|
|      = S + .|
| . + o o|
| . . o|
| . +o|
| o+++|
+---[SHA256]-----+
mci-hadoop@esbati-ASUSLaptop:~$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
mci-hadoop@esbati-ASUSLaptop:~$ chmod 0600 ~/.ssh/authorized_keys
mci-hadoop@esbati-ASUSLaptop:~$ ssh localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ED25519 key fingerprint is SHA256:EOcNuhF1fA56Nn6Hff4Ve1dMSHigynMZLLFcGLiD0vI.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'localhost' (ED25519) to the list of known hosts.
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.8.0-45-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

11 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

mci-hadoop@esbati-ASUSLaptop:~$
```

چرا نیاز به SSH بدون رمز عبور داریم؟

۱. اتوماسیون ارتباطات بین سرورها: در محیط‌هایی مانند Hadoop، که دارای معماری توزیع شده است، نودهای مختلف (ماشین‌ها یا سرورها) باید به‌طور مکرر و خودکار با یکدیگر ارتباط برقرار کنند. برای مثال، DataNode ها و NameNode ها باید بتوانند بدون نیاز به وارد کردن رمز عبور از طریق SSH با یکدیگر تبادل داده کنند.

۲. اجرای دستورات از راه دور: در Hadoop و سایر سیستم‌های توزیع شده، نیاز است که از طریق SSH بتوانیم از راه دور دستورات را روی نودهای دیگر اجرا کنیم. Passwordless SSH امکان می‌دهد که دستورات از طریق SSH به‌طور خودکار و بدون نیاز به وارد کردن رمز عبور اجرا شوند.

۳. سهولت در مدیریت خوشه‌ها (Clusters): در زمانی که مدیریت یک خوشه شامل چندین نود است، وارد کردن مکرر رمز عبور برای هر عملیات بین نودها بسیار زمان‌بر و دست‌وپاگیر خواهد بود. SSH بدون رمز عبور این فرآیند را تسهیل می‌کند.

مرحله بعد: (اجرای فرمان `hdfs namenode -format`)

فرمت کردن **NameNode** در Hadoop یکی از مراحل اولیه در راه‌اندازی سیستم فایل توزیع شده Hadoop (HDFS) است و برای آماده‌سازی NameNode جهت مدیریت سیستم فایل توزیع شده به کار می‌رود. در این مرحله، متادیتای سیستم فایل HDFS ایجاد و پیکربندی می‌شود تا HDFS بتواند کار خود را به درستی شروع کند.

مفهوم و اهمیت فرمت کردن NameNode

۱. NameNode چیست؟

- **NameNode** یک جزء مرکزی در HDFS است که وظیفه نگهداری و مدیریت متادیتای سیستم فایل توزیع شده را برعهده دارد. متادیتا شامل اطلاعاتی مانند ساختار دایرکتوری‌ها، فایل‌ها، و محل بلوک‌های داده در DataNode ها است.

- به عبارت ساده، **NameNode** به نوعی "نقشه" سیستم فایل است که نشان می‌دهد فایل‌های مختلف چگونه در بلوک‌های مختلف در سراسر خوشه توزیع شده‌اند.

۲. چرا باید **NameNode** را فرمت کنیم؟

- فرمت کردن **NameNode** یک مرحله اولیه برای راه‌اندازی **HDFS** است. این عملیات برای آماده‌سازی **NameNode** و ساخت ساختارهای مورد نیاز متادیتای سیستم فایل استفاده می‌شود. بدون این مرحله، **NameNode** نمی‌تواند فایل‌ها را مدیریت کرده یا ارتباط صحیحی با **DataNode**ها برقرار کند.
- در طی این فرایند، دایرکتوری‌های ذخیره‌سازی متادیتا که توسط **NameNode** استفاده می‌شود (مثل **dfs.name.dir** که در فایل **hdfs-site.xml** مشخص شده) فرمت می‌شوند و فایل‌های سیستمی پایه مورد نیاز **HDFS** در آنها ایجاد می‌شود.

چه زمانی این کار انجام می‌شود؟

- برای اولین بار پس از نصب **Hadoop** زمانی که برای اولین بار یک خوشه **Hadoop** راه‌اندازی می‌کنید، فرمت کردن **NameNode** ضروری است تا **NameNode** بتواند متادیتای **HDFS** را از ابتدا بسازد.
- در هنگام بازسازی یا ریست سیستم فایل: اگر قصد داشته باشید یک سیستم فایل توزیع‌شده جدید را از ابتدا ایجاد کنید یا **NameNode** به دلایلی خراب شده باشد و نیاز به راه‌اندازی دوباره باشد، باید آن را فرمت کنید.

و در نهایت اجرا دستورات:

```
start-dfs.sh
start-yarn.sh
```

```

esbati@esbati-ASUSLaptop:~$ start-dfs.sh
Starting namenodes on [localhost]
localhost: Warning: Permanently added 'localhost' (ED25519) to the list of known hosts.
Starting datanodes
localhost: Warning: Permanently added 'localhost' (ED25519) to the list of known hosts.
Starting secondary namenodes [esbati-ASUSLaptop]
esbati-ASUSLaptop: Warning: Permanently added 'esbati-asuslaptop' (ED25519) to the list of known hosts.
esbati@esbati-ASUSLaptop:~$ start-yarn.sh
Starting resourcemanager
Starting nodemanagers
localhost: Warning: Permanently added 'localhost' (ED25519) to the list of known hosts.
esbati@esbati-ASUSLaptop:~$ jps
323442 NameNode
324045 ResourceManager
323832 SecondaryNameNode
324552 Jps
324186 NodeManager
323579 DataNode
esbati@esbati-ASUSLaptop:~$ 

```

حال برای انتقال فایل های لاگ ایجاد شده در مرحله قبل کافی است، مراحل زیر را طی کنیم:

```

# Create HDFS directory
hdfs dfs -mkdir /log

# Move all log files to HDFS
hdfs dfs -put log-*.log /log

# Verify files in HDFS
hdfs dfs -ls /log

```

و در نهایت:

```

esbati@esbati-ASUSLaptop:~/mci-hadoop$ ./log_rotation.sh
esbati@esbati-ASUSLaptop:~/mci-hadoop$ hdfs dfs -ls /log
esbati@esbati-ASUSLaptop:~/mci-hadoop$ ls
file.log          log-error.log     log-info.log      log-warning-4.log
log-error-1.log   log-info-1.log    log_rotation.sh   log-warning-5.log
log-error-2.log   log-info-2.log    log-warning-1.log  log-warning.log
log-error-3.log   log-info-3.log    log-warning-2.log
log-error-4.log   log-info-4.log    log-warning-3.log
esbati@esbati-ASUSLaptop:~/mci-hadoop$ hdfs dfs -put log-*.log /log
esbati@esbati-ASUSLaptop:~/mci-hadoop$ hdfs dfs -ls /log
Found 16 items
-rw-r--r-- 1 esbati supergroup 1048636 2024-10-08 04:06 /log/log-error-1.log
-rw-r--r-- 1 esbati supergroup 1048613 2024-10-08 04:06 /log/log-error-2.log
-rw-r--r-- 1 esbati supergroup 1048609 2024-10-08 04:06 /log/log-error-3.log
-rw-r--r-- 1 esbati supergroup 1048591 2024-10-08 04:06 /log/log-error-4.log
-rw-r--r-- 1 esbati supergroup 387432 2024-10-08 04:06 /log/log-error.log
-rw-r--r-- 1 esbati supergroup 1048609 2024-10-08 04:06 /log/log-info-1.log
-rw-r--r-- 1 esbati supergroup 1048588 2024-10-08 04:06 /log/log-info-2.log
-rw-r--r-- 1 esbati supergroup 1048602 2024-10-08 04:06 /log/log-info-3.log
-rw-r--r-- 1 esbati supergroup 1048604 2024-10-08 04:06 /log/log-info-4.log
-rw-r--r-- 1 esbati supergroup 738632 2024-10-08 04:06 /log/log-info.log
-rw-r--r-- 1 esbati supergroup 1048620 2024-10-08 04:06 /log/log-warning-1.log
-rw-r--r-- 1 esbati supergroup 1048606 2024-10-08 04:06 /log/log-warning-2.log
-rw-r--r-- 1 esbati supergroup 1048603 2024-10-08 04:06 /log/log-warning-3.log
-rw-r--r-- 1 esbati supergroup 1048620 2024-10-08 04:06 /log/log-warning-4.log
-rw-r--r-- 1 esbati supergroup 1048617 2024-10-08 04:06 /log/log-warning-5.log
-rw-r--r-- 1 esbati supergroup 241050 2024-10-08 04:06 /log/log-warning.log
esbati@esbati-ASUSLaptop:~/mci-hadoop$ 

```